

300 PROGRAMAS RESUELTOS en BASIC

E. Lowy Frutos – A.E. Gallego Palomero
S. Mansilla Romo



EQUIPO DE AUTORES

Ernesto Lowy Frutos
Agregado de Física y Química de I.B.

A. Enrique Gallego Palomero
Agregado de Matemáticas de I.B.

Serafín Mansilla Romo
Licenciado en Matemáticas

EQUIPO EDITORIAL

Coordinación: José Luis Robles Cid

Maqueta: Angela Medina

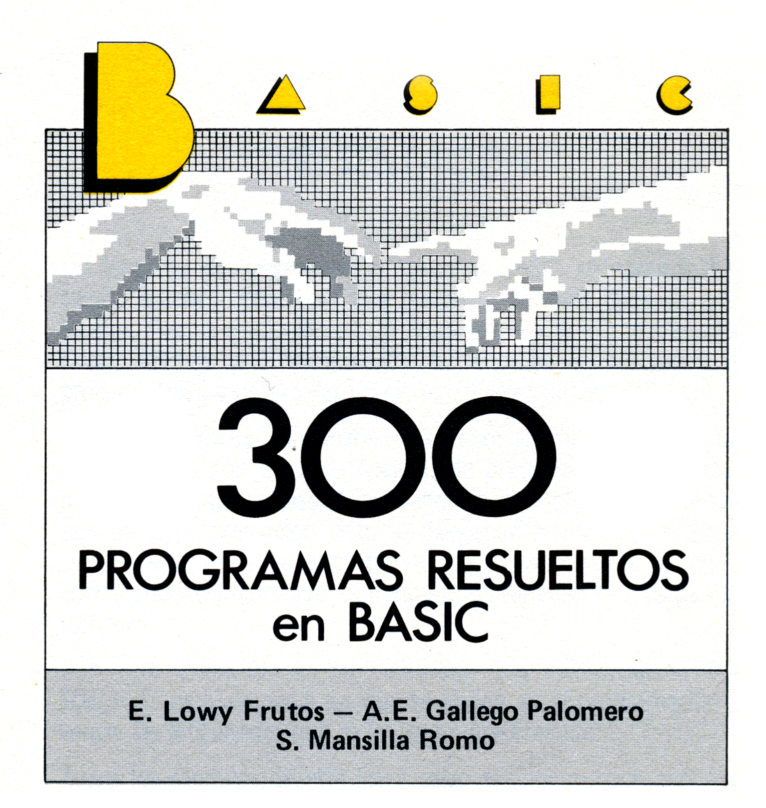
Dibujos: Luis Rojas

Fotografías: José Manuel Navia, AISA

Portada: Alfonso Ruano y José Luis Cortés

CARLOS CANO TURIEL

(VIÑUELEZ)



I.S.B.N.: 84-348-1720-9 / Depósito legal: M. 33.116-1985 / Fotocomposición: Grafilia, S. L.
Imprime: G. VELASCO, S. A. - Antonio de Cabezón, 13 - 28034 Madrid
Impreso en España - Printed in Spain

PRESENTACIÓN

Un lenguaje de programación como el BASIC se aprende haciendo muchos programas que resuelvan distintos tipos de problemas. Pero la resolución de un problema con el microordenador exige, además del conocimiento del lenguaje BASIC, seguir el proceso lógico y desarrollar un cierto esfuerzo de imaginación.

Ahora bien, en los comienzos no siempre se consigue enfrentar dicho desafío con éxito, lo cual puede producir un cierto desánimo y, en algunos casos, un abandono prematuro del aprendizaje de la programación. Esto se puede evitar pidiendo orientación a una persona que domine la programación, o bien, consultando un libro que contenga gran variedad de programas resueltos.

Con esta intención se ha elaborado *300 PROGRAMAS RESUELTOS:* para ayudar a los que se inician en la programación a vencer las dificultades que se les vayan presentando.

Todos los capítulos de este libro, salvo el primero, que contiene una breve introducción sobre el microordenador y el lenguaje BASIC, comprenden tres apartados:

- En el primero, RESUMEN DE LAS INSTRUCCIONES, se incluye un resumen de cada una de las instrucciones que se van a utilizar por primera vez en los programas del capítulo, acompañado de uno o más ejemplos. Éstos ayudan a afianzar la comprensión de cada una de las instrucciones, ya que permiten observar su funcionamiento en situaciones concretas.
- En el segundo apartado, PROGRAMAS RESUELTOS EXPLICADOS, contiene una serie de programas resueltos con la explicación del comportamiento de sus instrucciones.

- El tercer apartado, PROGRAMAS RESUELTOS, contiene otra serie de programas resueltos, pero sin explicación. Ésta se ha omitido por sobreentender que el mismo lector puede entenderlos apoyándose en las explicaciones del apartado anterior.

Los enunciados de los ejercicios de este libro coinciden con los del primer libro de la serie, *BASIC. Programación.* En este sentido, *300 PROGRAMAS RESUELTOS* puede considerarse como un complemento del primero, en cuanto que refuerza y profundiza su vertiente práctica. No obstante, cabe advertir que, aun teniendo en cuenta este carácter de complementariedad, ambos libros pueden utilizarse en forma independiente.

LOS AUTORES

Índice

1 Microordenadores

Lenguaje BASIC

1. El microordenador

- Observando a simple vista un microordenador se advierte que su aspecto externo es similar al de una máquina de escribir. Sus teclados únicamente se diferencian en unas pocas teclas. Sin embargo, ambas máquinas son completamente diferentes tanto en su constitución como en su funcionamiento.



Aspecto externo de un microordenador.

En realidad, un microordenador es un tipo de ordenador que cumple muchas de las funciones realizadas por éste y que, gracias a las tecnologías de miniaturización electrónica, ocupa un reducidísimo espacio.



- Para poner en funcionamiento el microordenador es necesario conectarlo a la red eléctrica y a un monitor o televisor.

  Si se conecta a un televisor es necesario seleccionar el canal 36 de UHF.

2. Partes de un microordenador

- El teclado y la pantalla son partes importantes del microordenador, pero lo esencial del mismo se encuentra en su interior: es la MEMORIA y el MICROPROCESADOR.

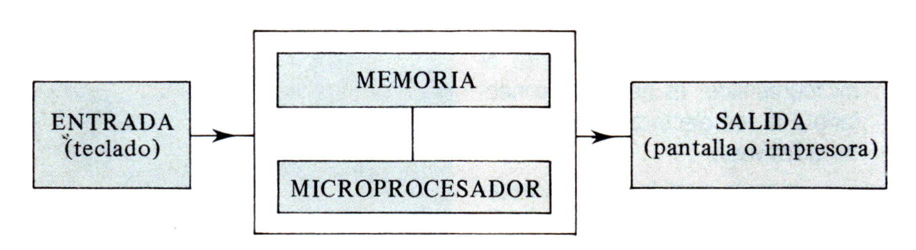
TECLADO

MEMORIA

MICROPROCESADOR

PANTALLA

En resumen, un microordenador se compone de cuatro partes unidas entre sí: la ENTRADA (teclado), la MEMORIA, el MICROPROCESADOR y la SALIDA (pantalla).



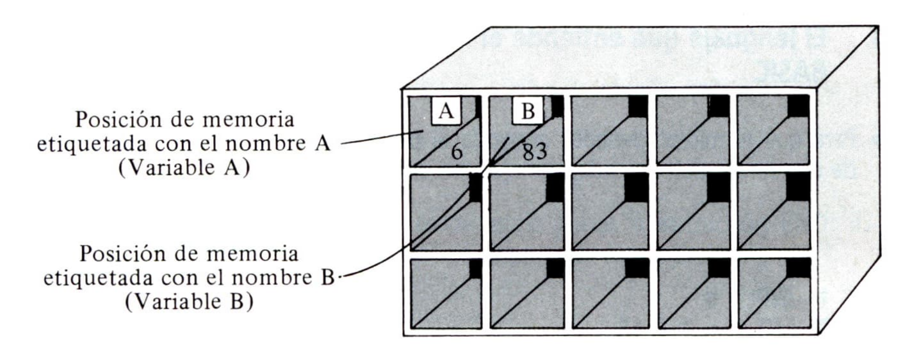
El dispositivo de salida en lugar de una pantalla (o monitor) puede ser una impresora.

- El **microprocesador** es el «cerebro» del microordenador: controla su funcionamiento, realiza operaciones aritméticas y lógicas, etc.
- La **memoria** almacena toda la información que necesita el microordenador para funcionar.

La memoria se puede comparar con un panal con muchas celdillas, llamadas **posiciones de memoria.**

Al introducir un dato en la memoria es necesario dar un nombre a la posición de memoria en la que se guarda dicho dato, pues de lo contrario sería imposible encontrarlo posteriormente.

Una posición de memoria etiquetada con un nombre se denomina variable, ya que puede contener datos diferentes (variables) en el curso de la ejecución de un programa.



La posición de memoria etiquetada con A almacena en este instante el dato numérico 6 y la etiquetada con B contiene el dato numérico 83.

Esto suele indicarse así:

A = 6
B = 83

(El nombre de la variable se escribe a la izquierda y el valor que tiene en un determinado instante se escribe a la derecha, separados por el signo igual.)

3. El programa

Para que el microordenador realice un proceso hay que indicarle el orden de los distintos pasos que tiene que seguir. Este conjunto de órdenes o instrucciones

constituye el PROGRAMA de trabajo al cual se ajustará ciegamente el microordenador.

Por ejemplo, si el proceso consiste en calcular y escribir en la pantalla el valor de A que se obtiene en la expresión

$$A = B + 15$$

para cualquier valor de la variable B introducido por el teclado, el orden de las instrucciones podría ser éste:

1 INTRODUCIR POR EL TECLADO EL VALOR DE B
2 CALCULAR EL VALOR DE A EN LA EXPRESION A = B + 15
3 ESCRIBIR O IMPRIMIR EN LA PANTALLA EL VALOR DE A
4 TERMINAR (FIN)

Este conjunto de órdenes o instrucciones constituye un ejemplo de PROGRAMA.

Para obtener el resultado de la suma y su escritura hay que ordenar al microordenador que **ejecute** el programa.

4. El lenguaje que entiende el microordenador: el lenguaje BASIC

- Para que el microordenador entienda el programa anterior debe estar escrito de una forma determinada, por ejemplo, así:

```
1 INPUT B
2 LET A = B + 15
3 PRINT A
4 END
```

El lenguaje en el que están escritas estas últimas instrucciones se llama BASIC, palabra que se forma con las iniciales de las palabras de la expresión inglesa «**B**iginner's **A**ll-purpose **S**imbolic **I**nstruction **C**ode» (Código de instrucción simbólica para principiantes para todos los usos). Este lenguaje consta de una serie de palabras inglesas y símbolos gráficos, que deben seguir unas reglas gramaticales estrictas.

- Vemos que el programa consta de un conjunto de líneas numeradas, que corresponden a las órdenes o instrucciones, y cada una de ellas está constituida por:
  - — una **palabra clave** que entiende el microordenador **(INPUT, LET, PRINT, END).**
  - — **variables,** como A y B, **números** y **operadores aritméticos** (+).
- Las instrucciones se pueden ordenar con los números 1, 2, 3, ..., pero se aconseja numerarlas de 10 en 10, pues ello permite añadir en cualquier momento instrucciones olvidadas, con numeración intermedia.

  Siguiendo esta norma, el programa anterior quedaría así:

```
10 INPUT B
20 LET A = B + 15
30 PRINT A
40 END
```

En algunos microordenadores no es necesario poner la instrucción **END;** incluso el microordenador ZX *Spectrum* no dispone de esta instrucción.

5. Cómo se almacena un programa en memoria

Una vez escrito un programa en el papel, se teclea cada instrucción pulsando a continuación la tecla [ENTER] (o [RETURN] o [NEW LINE], etc., según el microordenador).

En este libro utilizaremos [ENTER].

Cuando se acciona esta tecla la instrucción pasa automáticamente a la memoria. Si no se acciona, el microordenador no se da por enterado de la instrucción escrita; es decir, no pasa a la memoria. Por ejemplo, si se teclea

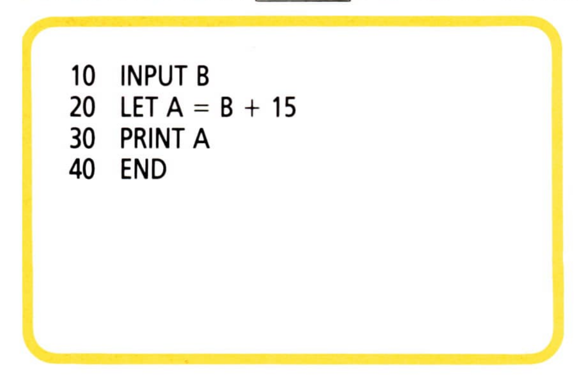
```
10 INPUT B
```

y no se pulsa la tecla [ENTER], la instrucción queda escrita en la pantalla pero no pasa a la memoria. En cambio, si se teclea

```
10 INPUT B [ENTER]
```

la instrucción aparece en la pantalla y, además, se almacena en la memoria.

Si tecleamos el programa anterior, lo veremos así en la pantalla. (¡NO OLVIDARSE DE PULSAR LA TECLA ENTER DESPUÉS DE CADA INSTRUCCIÓN!)



6. Cómo se ejecuta un programa almacenado en memoria

El programa lo tenemos ya almacenado en la memoria. ¿Qué tenemos que hacer para que el microordenador lo ejecute, es decir, para que interprete las instrucciones y realice los cálculos?

Para que el microordenador ejecute el programa se escribe la palabra **RUN**, pulsando a continuación la tecla ENTER.

Al ejecutar la primera instrucción

```
10 INPUT B
```

aparece en pantalla un signo de interrogación (en algunos microordenadores, **L**), deteniéndose momentáneamente la ejecución del programa.



Con esta interrogación (o con **L**) el microordenador pregunta qué valor deseamos asignar a la variable B de la primera instrucción. Si, por ejemplo, tecleamos **100** (y ENTER), la ejecución continúa, apareciendo inmediatamente en la pantalla el valor correspondiente a A:

```
115
```

¿Cómo ha llegado el microordenador al resultado 115?

Con el valor 100 almacenado en la posición de memoria, representada por B, el microordenador ejecuta la instrucción

```
20 LET A = B + 15
```

Esta instrucción indica a la máquina que realice las operaciones aritméticas indicadas a la derecha del signo igual y el resultado lo almacene en la posición de memoria representada por A, situada a la izquierda del signo igual. Es decir, calcula 100 + 15, y el resultado 115 lo almacena en A.

A continuación se ejecuta la instrucción

```
30 PRINT A
```

que ordena al microordenador escriba o imprima en la pantalla el valor almacenado en la posición de memoria representada por A.

Finalmente, se ejecuta la instrucción

```
40 END
```

que indica al microordenador que el programa ha terminado.

7. Cómo se cambian las instrucciones de un programa y cómo se borran

- Según se va escribiendo una instrucción de un programa es normal que se teclee un símbolo por otro. Si uno se da cuenta del error cometido antes de pulsar la tecla [ENTER], es decir, antes de cargar en memoria la instrucción, se puede corregir el error utilizando una tecla que tienen todos los microordenadores y, aunque recibe distinto nombre, cumple la misma función: hace retroceder el cursor, borrando, uno a uno, los símbolos escritos.
- Si deseamos hacer algún cambio en una determinada instrucción de un programa cargado ya en la memoria procedemos de la siguiente manera:

— Averiguamos primero el número de la instrucción o instrucciones que deseamos modificar. Para ello se hace aparecer en la pantalla el listado del programa, tecleando la instrucción **LIST** (y [ENTER]).

Supongamos que en este momento está cargado en la memoria del microordenador el programa anterior. Utilizando la instrucción **LIST** se obtiene su listado en la pantalla:

```
10  INPUT B
20  LET A = B + 15
30  PRINT A
40  END
```

— Si ahora deseamos modificar la instrucción **20**, por ejemplo, basta teclearla de nuevo, introduciendo las modificaciones. Si escribimos

```
20  LET P = 0.5 * B + 16
```

y pulsamos la tecla [ENTER], se borra de la memoria la primitiva instrucción **20** y queda cargada la **20** modificada. Para verificarlo, listamos de nuevo el programa y obtendremos en pantalla el nuevo listado:

```
10  INPUT B
20  LET P = .5 * B + 16  (*)
30  PRINT A
40  END
```

- Cuando se conoce de antemano el número de la instrucción que se desea modificar, no es necesario listar el programa; es suficiente con teclearla de nuevo. Sin embargo, para mayor seguridad conviene listarla, para lo cual se teclea la instrucción **LIST** y, a continuación, el número de la instrucción (y [ENTER]). Así, tecleando **LIST 20** (y [ENTER]) se consigue hacer aparecer en la pantalla la instrucción 20. Para modificarla, ésta se teclea de nuevo.

(*) Los números decimales con parte entera nula, el microordenador los escribe omitiendo el cero.

- Si se desea eliminar una instrucción de un programa basta con teclear el número de la misma (y [ENTER]). Por ejemplo, si escribimos 30 (y [ENTER]) y, a continuación, listamos el programa anterior, se obtiene lo siguiente en la pantalla:

```
10  INPUT B
20  LET P = .5 * B + 16
40  END
```

En este listado comprobamos que la instrucción **30 PRINT A** no se encuentra ya cargada en la memoria.

- Para borrar todo el contenido de la memoria se escribe **NEW** (nuevo programa) y se pulsa a continuación la tecla [ENTER].

  También se puede borrar todo el contenido de la memoria desconectando y conectando el microordenador.

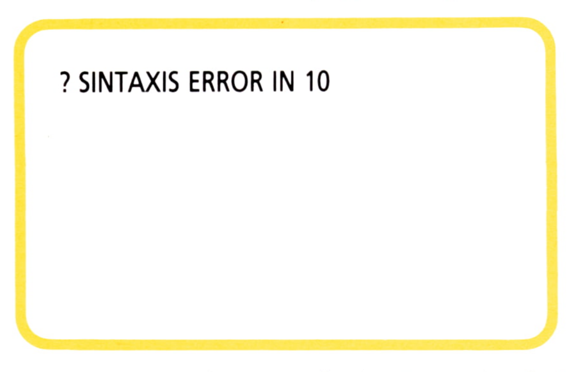
- Para borrar solamente lo que está escrito en la pantalla, la mayoría de los microordenadores utiliza la instrucción **CLS.** Esta instrucción no borra lo almacenado en la memoria.

8. Errores de sintaxis

Tecleamos el siguiente programa que calcula el producto de un número X por 13.85 e imprime el resultado:

```
10  IMPUT X
20  LET Y = 13.85 * X
30  PRINT Y
40  END
```

Inmediatamente después de escribir **RUN** (y ENTER) para ejecutar el programa, el microordenador, en lugar de imprimir el resultado, nos informa que hemos cometido un error al teclear el programa, imprimiendo un mensaje en la pantalla:



Este mensaje significa *error de sintaxis en* **10.** Efectivamente, si nos fijamos bien en la instrucción **10** observamos que escribimos **IMPUT**, con **M**, en lugar de **INPUT.** (El mensaje de error puede ser diferente, según el microordenador; puede ser incluso un número clave.)

Para corregir este error se modifica la instrucción **10** mediante el procedimiento explicado en el apartado anterior.

9. Resumen de las palabras BASIC estudiadas en este capítulo

Palabra	Función
INPUT (*)	Introduce por teclado el valor de una variable.
LET (*)	Calcula el valor de una expresión y lo asigna a una variable.
PRINT (*)	Escribe en la pantalla el valor de una variable.
END	Indica la terminación de un programa.
ENTER, RETURN, u otra equivalente	Hace que el dato o instrucción tecleados se almacene en la memoria.
RUN	Inicia la ejecución de un programa. (En el argot informático suele decirse «correr» el programa.)

(*) Las instrucciones **INPUT, LET** y **PRINT** se estudiarán con más detalle en los capítulos 2 y 3.

Palabra	Función
LIST LIST n LIST $n_1 - n_2$ LIST n – LIST – n	Lista el programa almacenado en la memoria. Lista la instrucción numerada con **n.** Lista la serie de instrucciones comprendidas entre **n_1** y **n_2**, éstas inclusive. Lista las instrucciones comprendidas entre **n** y el final. Lista las instrucciones anteriores a la **n.**
NEW	Borra todo el contenido de la memoria.
CLS	Borra lo que está escrito en la pantalla, pero no el contenido de la memoria.

2 Comenzando a programar en BASIC

Primeros programas resueltos con INPUT, PRINT, LET y END

RESUMEN DE LAS INSTRUCCIONES

INPUT

Esta instrucción permite introducir datos desde el teclado, los cuales se asignan a las variables.



Ejemplo

Cuando el microordenador ejecuta

```
10 INPUT R
```

aparece en la pantalla una **?** (o una **L**), esperando a que se teclee un valor numérico, el cual quedará asignado a la variable R.

Después de tecleado el valor, continúa la ejecución de las otras instrucciones del programa.

PRINT

Esta instrucción imprime en la pantalla los valores de las variables.



Ejemplo

Cuando el microordenador ejecuta la instrucción

```
50  PRINT L
```

inscribe en la pantalla el valor de la variable L. Si, por ejemplo, en este instante, el valor de L es 23.207, al ejecutarse **50 PRINT L** aparece en la pantalla 23.207.

LET

La instrucción LET **asigna** a una variable un valor o un resultado de una expresión aritmética, una vez calculada.



Ejemplos

- Al ejecutarse la instrucción

  30 LET A = 0.238

  a la variable A se asigna el valor 0.238.

- Cuando se ejecuta la instrucción

  50 LET L = 2 * PI * R

  a la variable L se asigna el resultado que se obtiene al calcular la expresión 2 * 3.14159265 * valor de R.

Observaciones

- En la mayoría de los microordenadores se puede suprimir la palabra **LET**; es decir, en lugar de

  50 LET L = 2 * PI * R

  se puede escribir

  50 L = 2 * PI * R

- En una expresión aritmética, primero se calculan las operaciones que están entre paréntesis, después las potencias (∧, ↑ ó **), después los productos (*) y cocientes (/), y finalmente, las sumas (+) y las restas (−).

  Una vez tenido en cuenta este orden de prioridad, la expresión se calcula de izquierda a derecha.

END

Esta instrucción indica el **fin** de un programa.



Observación

Algunos microordenadores no exigen incluir **END**. Incluso el ZX *Spectrum* no dispone de esta instrucción. En su lugar se puede poner **STOP**.

2.1. Escribe un programa que calcule la expresión Y = 0,25 X + 0,8, e imprima el resultado.

```
10 INPUT X
20 LET Y = 0.23 * X + 0.8
30 PRINT Y
40 END
```

— Mediante la instrucción **10** se introduce por el teclado un número que se asigna a X.
— La instrucción **20** calcula el valor de la expresión 0.23 * X + 0.8 para X = número introducido.
— La instrucción **30** escribe en la pantalla el valor de Y, calculado por la instrucción anterior.
— La instrucción **40** da por finalizado el programa.

2.2. Explica cómo harías para ejecutar el programa anterior.

¿Se detiene la ejecución en algún momento? ¿Qué tienes que hacer para que continúe la ejecución?

— Para ejecutar el programa expuesto anteriormente hay que teclear **RUN** [ENTER].
— La ejecución se detiene inmediatamente, apareciendo una ? o una L en la pantalla.
— Para que continúe la ejecución hay que introducir un valor por el teclado.

2.3. Escribe un programa que calcule la suma de dos números e imprima el resultado.

```
10 INPUT A
20 INPUT B
30 LET S = A + B
40 PRINT S
50 END
```

— Al ejecutarse este programa, la ejecución se detiene dos veces. En la primera espera a que se introduzca por el teclado un valor para A y, en la segunda, espera a que se introduzca un valor para B.
— La instrucción **30** calcula A + B, y el resultado lo asigna a S. El valor de S lo imprime la introducción **40.**

2.4. Escribe un programa que calcule el producto de dos números e imprima el resultado.

```
10  INPUT X
20  INPUT Y
30  LET P = X * Y
40  PRINT P
50  END
```

— Como en el programa anterior, las instrucciones **10** y **20** hacen que se pueda introducir por el teclado dos valores, uno para X y otro para Y.
— La instrucción **30** imprime en la pantalla el valor de P.

2.5. ¿Qué aparecerá en la pantalla al ejecutar el siguiente programa?

```
10  INPUT A
20  INPUT B
30  LET C = A ↑ B
40  PRONT C
50  END
```

Aparecerá un mensaje de este tipo:

SINTAXIS ERROR IN 40

debido a que la palabra **PRONT** está mal escrita. En lugar de **PRONT** es **PRINT**.

2.6. Explica cómo harías para corregir el programa anterior, de tal manera que al ejecutarlo llegue a imprimir el valor de C.

Tecleando de nuevo la instrucción **40**, escribiéndola correctamente:

```
40  PRINT C
```

2.7. Escribe un programa que calcule el cuadrado de un número e imprima el resultado.

```
10  INPUT N
20  LET C = N ↑ 2
30  PRINT C
40  END
```

— Mediante la instrucción **10** se introduce el valor de N.
— La instrucción **20** calcula el cuadrado de N y el resultado lo asigna a la variable C.
— La instrucción **30** imprime el valor de C.

2.8. Indica qué calcula el siguiente programa:

```
10  INPUT A
20  INPUT B
30  INPUT C
40  LET X = A * B * C
50  PRINT X
60  END
```

¿Cuántas veces se detiene la ejecución de este programa? ¿Por qué?

— Este programa calcula el producto de tres números.
— La ejecución del programa se detiene tres veces porque contiene tres instrucciones **INPUT**.

2.9. El siguiente programa debe sumar dos números y a la suma multiplicarla por 5.

¿Logra hacer este cálculo? En caso negativo modifica el programa.

```
10  INPUT H
20  INPUT J
30  LET P = H + J * 5
40  PRINT P
50  END
```

— No logra hacer el cálculo, ya que con la expresión H + J * 5, el 5 sólo multiplica a J.
— Para que el programa lograra hacer el cálculo correctamente habría que escribir la instrucción **30** así:

```
30  LET P = (H + J) * 5
```

PROGRAMAS RESUELTOS

2.1. Escribe el programa que calcule el producto de dos números.

```
10  INPUT A
20  INPUT B
30  LET P = A * B
40  PRINT P
50  END
```

2.2. Explica qué hace el siguiente programa:

```
10  INPUT A
20  LET B = A + 1
30  PRINT B
40  END
```

Este programa halla el número siguiente a A, y lo escribe en la pantalla.

2.3. Suponte que el programa del ejercicio anterior está cargado en la memoria del microordenador. ¿Qué tienes que hacer para ejecutarlo?

Teclear **RUN** [ENTER].

2.4. Según estás tecleando la instrucción

20 LEP B = A + 1

te das cuenta, antes de pulsar la tecla [ENTER], que has cometido un error al escribir **LEP** en lugar de **LET**. ¿Qué harías para corregir este error?

Para corregir este error se hace retroceder el cursor hasta que queda superpuesto con la letra P; a continuación se teclea T, y finalmente se pulsa [ENTER].

2.5. Al ejecutar el siguiente programa, aparece en la pantalla un mensaje de error. Averigua cuál es dicho error y el mensaje que emite el microordenador.

```
10  INPUT N
20  LET = C = N ↑ 3
30  PRINT C
40  END
```

— El error está en la instrucción **20,** y es debido al primer signo igual que está a la derecha de la palabra **LET.**
— El mensaje que aparece en la pantalla es similar a éste:
SINTAXIS ERROR IN 20.

2.6. Explica cómo harías para corregir el programa anterior para lograr que termine su ejecución.

Se corrige la instrucción **20** volviéndola a escribir correctamente:
20 LET C = N ↑ 3

2.7. Al teclear el programa siguiente tienes un lapsus y escribes una instrucción dos veces con número diferente. ¿Cómo harías para borrar la instrucción que no corresponde?

```
10 INPUT P
20 LET D = P/168.690
30 PRINT D
40 PRINT D
50 END
```

Escribiendo lo siguiente:

40 ENTER

2.8. Después de escribir un programa en el papel deseas cargarlo en memoria, estando almacenado en ésta un programa que no te interesa conservarlo. ¿Qué harías antes de empezar a teclear el nuevo programa?

Antes de empezar a teclear el nuevo programa hay que teclear **NEW** [ENTER] para borrar todo el contenido de la memoria.

2.9. Escribe en lenguaje BASIC la siguiente expresión:

$$y = 0{,}27\,(x^3 - \frac{1}{2}\,x + \frac{1}{25})^2 : 30 - 25$$

LET Y = 0.27 * (X ↑ 3 − 1/2 * X + 1/25) ↑ 2/30 − 25

2.10. En los programas siguientes indica en qué orden se hacen las operaciones y qué resultado imprimirá el microordenador en la pantalla.

a)
```
10 LET Y = 5 − 5 * 8 ↑ 2
20 PRINT Y
30 END
```

— En la instrucción **10**, primero se calcula 8 ↑ 2; a este resultado se multiplica por 5, y finalmente, al primer 5 se resta el resultado anterior.
— En la pantalla aparece este resultado: −315.

b)
```
10 LET Y = (5 − 5) * 8 ↑ 2
20 PRINT Y
30 END
```

— Primero se hace la resta (5 − 5); después se calcula 8 ↑ 2 y finalmente se multiplican ambos resultados.
— En la pantalla aparece 0, como resultado final.

c) **10 LET Y = 5 − 2 * 4 ↑ 2**
20 PRINT Y
30 END

— En la instrucción **10,** primero se calcula 4 ↑ 2; a este resultado se multiplica por 2, y finalmente, al 5 se resta el resultado anterior.
— En la pantalla se imprime −27.

d) **10 LET Y = 5 − (2 * 4) ↑ 3**
20 PRINT Y
30 END

— Primero se hace el producto (2 * 4), después este producto se eleva al cubo, y finalmente, al 5 se resta el resultado anterior.
— En la pantalla aparece escrito −507.

3 Avanzando en la programación

Programas resueltos con INPUT, REM y PRINT

RESUMEN DE LAS INSTRUCCIONES

INPUT

Esta instrucción permite introducir un texto entrecomillado y uno o varios datos, los cuales se asignan a las variables correspondientes.

El texto entrecomillado se separa de las variables mediante un *punto y coma,* y las variables entre sí, mediante *comas.*



Ejemplos

- Cuando el microordenador ejecuta esta instrucción

```
20 INPUT "RADIO"; R
```

imprime en la pantalla la palabra RADIO y después interrumpe la ejecución hasta que se introduzca por el teclado un valor para la variable R.

- Si la instrucción es

```
30 INPUT "RADIOS"; R1, R2
```

el microordenador imprime en la pantalla la palabra RADIOS, e interrumpe la ejecución hasta que se introduzca por el teclado un valor para la variable R1. Una vez introducido, se interrumpe otra vez la ejecución hasta que se introduzca otro valor para la variable R2.

REM

Esta instrucción permite añadir **aclaraciones** o **comentarios** en un programa sin afectar a su ejecución.

La instrucción **REM,** aunque no tiene ninguna influencia en la ejecución de un programa, sí ocupa posiciones de memoria.

Ejemplo

Cuando el microordenador ejecuta un programa y se encuentra con una instrucción como ésta

55 REM CALCULA LA RAÍZ CUADRADA

pasa a la siguiente instrucción sin que se produzca ninguna alteración en la ejecución del programa.

Observar que el comentario no es necesario ponerlo entre comillas.

PRINT

La instrucción PRINT imprime en la pantalla los caracteres que se encuentran entre comillas y los valores de las variables.

Esta instrucción también puede calcular el valor de una expresión aritmética, para imprimirlo a continuación.

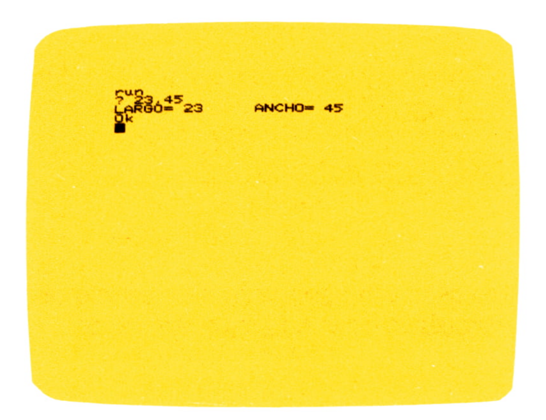


Ejemplos

- La ejecución de

  10 INPUT L, A
  20 PRINT "LARGO ="; L, "ANCHO ="; A

  imprime esta pantalla:



— El texto entrecomillado aparece en la pantalla sin comillas.

— En la pantalla se imprimen los valores de las variables L y A, pero no éstas.

— LARGO = y ANCHO = aparecen separados en la pantalla.

- El **punto y coma** y la **coma** en la instrucción **PRINT**.

  — Lo que está separado por un **punto y coma** se imprime **seguido** en la pantalla. Ejemplo: LARGO = y ... y lo mismo ANCHO y ...

  — Lo que está separado por una coma se imprime **separado,** en dos zonas contiguas de la pantalla. Ejemplo: LARGO = ... y ANCHO = ...

- Una instrucción **PRINT** sin variables ni textos deja una línea en blanco en la pantalla. Ejemplo: ejecutando este programa

```
10  INPUT A, B, C
20  PRINT A
30  PRINT B
40  PRINT
50  PRINT C
60  END
```

se obtiene esta pantalla



- Si una instrucción **PRINT** va seguida de una expresión aritmética, primero calcula el valor de dicha expresión y después lo imprime. Ejemplo: ejecutando

```
 5  PI = 3.1416
10  INPUT R
20  PRINT 2 * PI * R
```

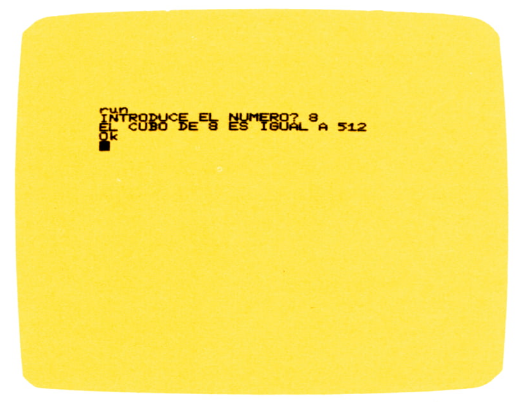
se obtiene en la pantalla el valor de la longitud de la circunferencia para el radio introducido.

PROGRAMAS RESUELTOS EXPLICADOS

3.1. Escribe un programa que calcule el cubo de un número e imprima el resultado. Utiliza la instrucción **REM** para dar título al programa.

```
10 REM CALCULA EL CUBO DE UN NUMERO
20 INPUT "INTRODUCE EL NUMERO"; N
30 LET C = N ↑ 3
40 PRINT "EL CUBO DE"; N; "ES IGUAL A"; C
50 END
```

- La instrucción **10** da título al programa sin producir ninguna alteración en la ejecución del mismo.
- La instrucción **20** introduce el valor de N y la **30** calcula el cubo C.
- La instrucción **40** imprime en la pantalla lo que muestra esta pantalla.



3.2. Escribe un programa que calcule la longitud de la circunferencia (L = 2 × 3,1416 R) y el área del cuadrado ($S = 1^2$), utilizando dos veces la instrucción REM, e imprime los resultados.

```
10 REM LONGITUD DE LA CIRCUNFERENCIA
15 PI = 3.1416
20 INPUT "RADIO"; R
30 LET L = 2 * PI * R
40 PRINT "LONG. CIRC. ="; L
50 REM AREA DEL CUADRADO
60 INPUT "LADO"; L1
70 LET A = L1 ↑ 2
80 PRINT "AREA CUAD. ="; A
90 END
```

- Las instrucciones **10** y **50** dan título a las dos partes del programa sin alterar su ejecución.

— Después de introducir un valor para R en la instrucción **20,** la **30** calcula L, y la **40** imprime
LONG. CIRC. = (valor de L)

— La ejecución del programa se interrumpe en la instrucción **60,** esperando a que por el teclado se introduzca un valor para L1.

— La instrucción **70** calcula el valor de A y la **80** imprime
AREA CUAD. = (valor de A)

3.3. Escribe lo que aparecerá impreso en la pantalla al ejecutarse los siguientes programas:

a)
```
10 LET X = 3.25
20 PRINT "X"
30 END
```

b)
```
10 LET X = 3.25
20 PRINT X
30 END
```

c)
```
10 LET X = 3.25
20 PRINT "X="; X
30 END
```

a) Ejecutando
```
10 LET X = 3.25
20 PRINT "X"
30 END
```
se obtiene en la pantalla la letra X, pero no el valor de X.

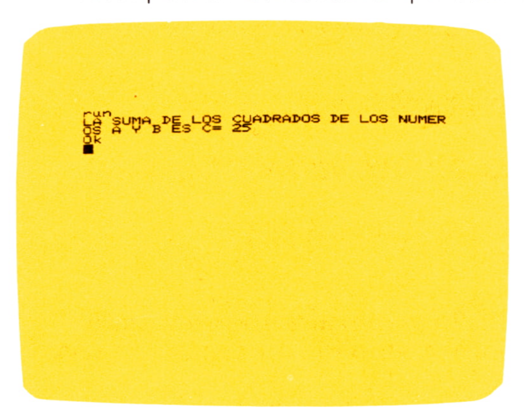
b) Ejecutando
```
10 LET X = 3.25
20 PRINT X
30 END
```
se obtiene en la pantalla el valor de X, es decir, 3.25.

c) Ejecutando
```
10 LET X = 3.25
20 PRINT "X="; X
30 END
```
en la pantalla aparece X = 3.25.

3.4. Escribe lo que imprimiría en la pantalla la instrucción **40** una vez ejecutada:

```
10 LET A = 3
20 LET B = 4
30 LET C = A ↑ 2 + B ↑ 2
40 PRINT "LA SUMA DE LOS CUADRADOS DE LOS NUMEROS A
   Y B ES C="; C
50 END
```

En la pantalla se obtiene lo que muestra la fotografía.



3.5. Escribe un programa que calcule el producto de un número X por 5, almacene el resultado en P e imprima en una línea:

EL NUMERO ES X = (valor de X)

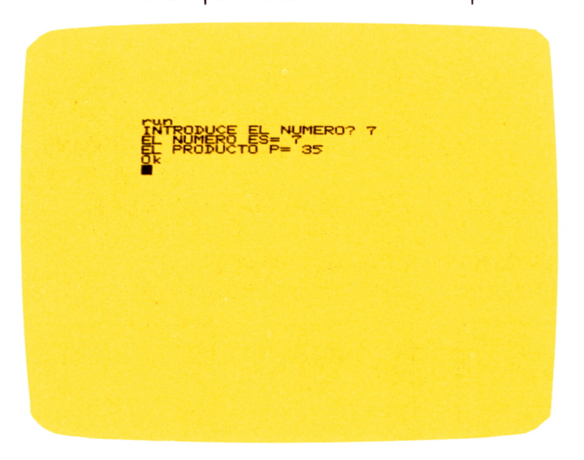
y en otra línea:

EL PRODUCTO P = (valor de P).

```
10  REM PRODUCTO DE UN NUMERO POR 5
20  INPUT "INTRODUCE EL NUMERO"; X
30  PRINT "EL NUMERO ES ="; X
40  LET P = X * 5
50  PRINT "EL PRODUCTO P ="; P
60  END
```

— La instrucción **20** permite introducir un valor para X.

— Las instrucciones **30** y **50** imprimen los valores de X y de P, acompañados de los textos que están entre comillas.



3.6. Elabora un programa que calcule el área del círculo (S = 3,1416 R^2) utilizando instrucciones PRINT para presentar con claridad el resultado en la pantalla.

```
10  REM SUPERFICIE DEL CIRCULO
15  PI = 3.1416
20  INPUT "INTRODUCE EL VALOR DEL RADIO"; R
30  LET S = PI * R ↑ 2
40  PRINT "PARA R = "; R; "EL VALOR DE S = "; S
50  END
```

— Mediante la instrucción **20** se introduce el valor del radio.
— La instrucción **30** calcula la superficie del círculo y su valor lo asigna a la variable de S.
— La instrucción **40** imprime los valores de R y de S, acompañados de los textos que están entre comillas. Si se introduce 10 para R, se obtiene este resultado en la pantalla:
PARA R = 10 EL VALOR DE S = 314.16

PROGRAMAS RESUELTOS

3.1. ¿Cómo aparecerán escritos los números 5 y 37 al ejecutarse los siguientes programas?

a)
```
10 PRINT 5, 37
20 END
```

b)
```
10 PRINT 5; 37
20 END
```

c)
```
10 PRINT 5
20 PRINT 37
30 END
```

d)
```
10 PRINT 5
20 PRINT
30 PRINT
40 PRINT 37
50 END
```

En una pantalla de dos zonas los números aparecerían así:

a)
```
5          37
```

b)
```
5 37
```

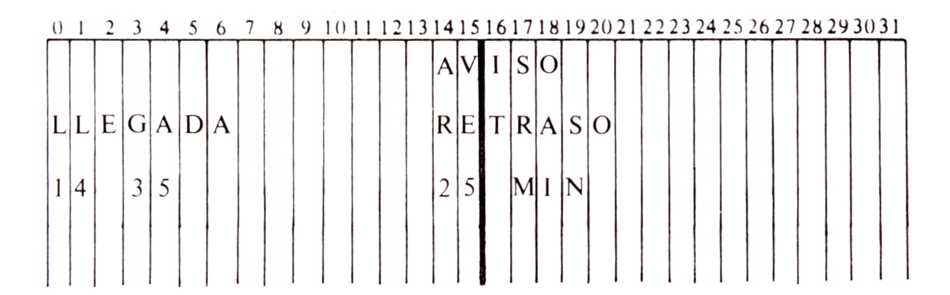
c)
```
5
37
```

d)
```
5

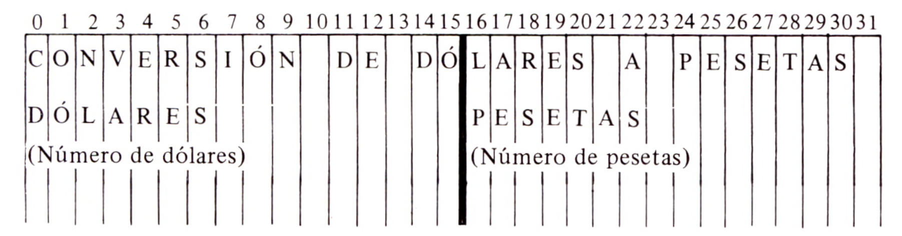

37
```

3.2. Escribe en un papel lo que aparecerá en la pantalla al ejecutarse el siguiente programa. (Suponte que se introduce 14.35 para H y 25 para R.)

```
 5  REM LLEGADA DE TRENES
10  INPUT H
20  INPUT R
30  PRINT "             AVISO"
           (13 espacios)
40  PRINT
50  PRINT "LLEGADA", "RETRASO"
60  PRINT H, R; "MIN"
70  END
```



3.3. Escribe un programa que haga la conversión de dólares a pesetas y escriba en la pantalla lo siguiente:




```
10  REM CONVERSION DE DOLARES A PESETAS
20  INPUT "DOLARES"; D
30  LET P = D*168.027
40  PRINT "CONVERSION DE DOLARES A PESETAS"
50  PRINT
60  PRINT "DOLARES", "PESETAS"
70  PRINT D, P
80  END
```

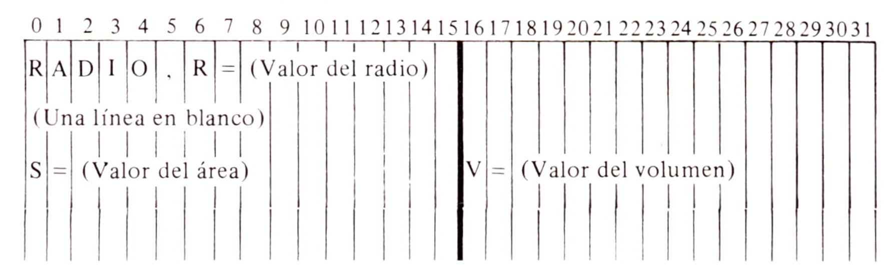
(Cotización: 1 dólar = 168.027 pesetas).

3.4. Averigua la cotización del día y escribe un programa que convierta pesetas a francos belgas. Incluye en el programa instrucciones **PRINT** que presenten los resultados en la pantalla en forma similar a la presentada en el ejercicio anterior.

```
10 REM CONVERSION DE PESETAS A FRANCOS BELGAS
20 INPUT "PESETAS"; P
30 LET FB =
40 PRINT "CONVERSION DE PESETAS A FRANCOS BELGAS"
50 PRINT
60 PRINT "PESETAS", "FRANCOS BELGAS"
70 PRINT P, FB
80 END
```

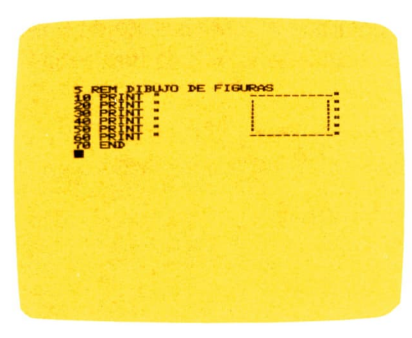
Cotización:

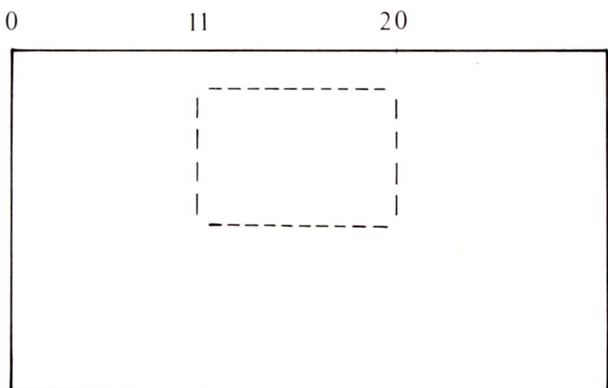
3.5. Las fórmulas del área y del volumen de la esfera son $S = 4 \pi R^2$ y $V \frac{3}{4} \pi R^3$, respectivamente. Elabora un programa que introduzca el valor del radio R, calcule S y V, y presente los resultados así:



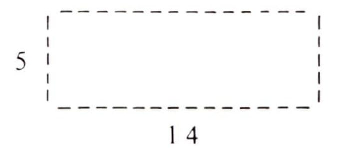

```
10 REM AREA Y VOLUMEN DE LA ESFERA
20 INPUT "RADIO DE LA ESFERA"; R
30 LET S = 4 * 3.14159265 * R ↑ 2
40 LET V = 4/3 * 3.14159265 * R ↑ 3
50 PRINT "RADIO, R = " ; R
60 PRINT
70 PRINT "S="; S, "V="; V
80 END
```

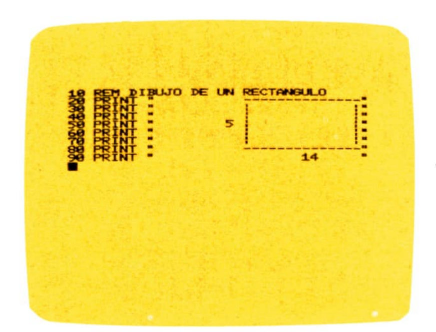
3.6. Dibuja en un papel la figura que hace aparecer en la pantalla el siguiente programa:





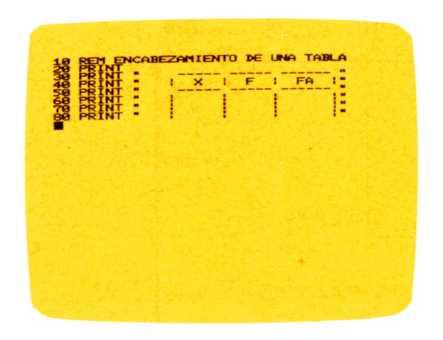
3.7. Haz un programa que dibuje en la pantalla la siguiente figura, incluidos los números que expresan la medida de los lados.





3.8. Realiza un programa que dibuje en la pantalla el encabezamiento de la siguiente tabla estadística; en la que aparezcan la variable X, la frecuencia F y la frecuencia acumulada, FA.

```
 ------  ------  -------
|   X   |   F   |   FA   |
 ------  ------  -------
|       |       |        |
|       |       |        |
|       |       |        |
```



3.9. La fórmula que calcula el interés simple I en función del capital C, el rédito R y el tiempo T, expresado en años, es:

$$I = \frac{CRT}{100}$$

Escribe un programa que calcule el interés y presente los datos y resultados en la pantalla así:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
C	=	(Valor de C)														R	=	(Valor de R)													
T	=	(Valor de T)														I	=	(Valor de I)													

```
10 REM INTERES SIMPLE
20 INPUT "INTRODUCE C, R, T"; C, R, T
30 LET I = C * R * T/100
40 PRINT
50 PRINT "C="; C; "     R="; R, "T="; T; "     I="; I
60 END
```

3.10. La fórmula que transforma grados centígrados a grados Fahrenheit es

$$F = \frac{9}{5} C + 32$$

Realiza un programa que haga la conversión de grados centígrados a Fahrenheit y presente en la pantalla el dato y resultado así:

GRADOS CENT. GRADOS FAHR.

(Una línea en blanco)

(Valor de gr. cent.) (Valor de gr. Fahr.)

```
10 REM CONVERSION DE GRADOS CENTIGRADOS A FAHRENHEIT
20 INPUT "GRADOS CENTIGRADOS"; C
30 LET F = 9/5 * C + 32
40 PRINT "GRADOS CENT.", "GRADOS FAHR."
50 PRINT
60 PRINT C, F
70 END
```

3.11. Las fórmulas de la altura descendida y de la velocidad alcanzada por un móvil que cae libremente en el vacío son:

$$h = \frac{1}{2} g t^2 \qquad y \qquad v = g t$$

donde h es la altura descendida, g la aceleración de la gravedad ($9,8 \frac{m}{s^2}$) y t el tiempo. Escribe un programa que calcule la altura descendida y la velocidad alcanzada para un tiempo t. Escribe instrucciones **PRINT** para presentar con claridad los datos y resultados en la pantalla.

```
10 REM CALCULO DE LA ALTURA Y VELOCIDAD EN LA CAIDA LIBRE
20 INPUT "TIEMPO"; T
30 LET H = 1/2 * 9.8 * T ↑ 2
40 LET V = 9.8 * T
50 PRINT "SI T="; T
60 PRINT "LA ALTURA DESCENDIDA, H ="; H
70 PRINT "Y LA VELOCIDAD ALCANZADA, V="; V
80 END
```

3.12. Recuerda la ley de Ohm que relaciona la resistencia R de un conductor, la caída de potencia V que se produce entre los extremos de la resistencia al pasar la intensidad I por la misma:

$$R = \frac{V}{I}$$

Haz un programa que calcule la intensidad I en función de R y V y presente los datos y resultados de la siguiente forma:

RESISTENCIA, R = (valor de R).
CAIDA DE TENSION, V = (valor de V).
INTENSIDAD, I = (valor de I).

```
10  REM LEY DE OHM
20  INPUT "INTRODUCE LOS VALORES DE V Y DE I"; V, I
30  LET R = V/I
40  PRINT "RESISTENCIA, R="; R
50  PRINT "CAIDA DE TENS, V="; V
60  PRINT "INTENSIDAD, I="; I
70  END
```

4 Saltos incondicionales e instrucciones condicionales

Programas resueltos con GOTO e IF ... THEN ... STEP

RESUMEN DE LAS INSTRUCCIONES

GOTO

Esta instrucción interrumpe la secuencia normal en la ejecución de un programa y transfiere el control a la línea cuyo número se especifica a la derecha de la palabra GOTO.



Ejemplo

Cuando en un programa, el microordenador ejecuta

```
80  GOTO 130
```

transfiere el control a la línea **130**, continuando la ejecución del programa a partir de esta línea.

Observaciones

La instrucción **GOTO** puede utilizarse también como **comando directo.** Así, **GOTO 90** (sin número delante) ejecuta, después de pulsar la tecla **ENTER** , el programa que se encuentra en memoria, a partir de la línea **90.**

Esta forma de iniciar la ejecución de un programa tiene la ventaja de conservar en la memoria los valores asignados previamente a variables en instrucciones **INPUT,** cosa que no ocurre cuando se inicia la ejecución con **RUN**.

IF ... THEN ...

El microordenador ejecuta la instrucción escrita a continuación de **THEN** si se cumple la condición expresada después de **IF**. En caso contrario, continúa la ejecución del programa en la instrucción siguiente a **IF ... THEN ...**



Ejemplos

- Cuando se ejecuta la instrucción

  50 IF A > 2 AND A < 5 THEN PRINT A

  si A es mayor que 2 y a la vez menor que 5, entonces se imprime el valor de A.
- La instrucción

  70 IF A < > B OR B < > C THEN GOTO 1000

  transfiere el control a la instrucción **1000** si A es distinto de B o B es distinto de C, y en caso contrario, la ejecución continúa en la instrucción siguiente.
- La instrucción

  90 IF A + 3 < B ↑ 2 AND C ↑ 2 > 3 THEN LET X = 2 * A

  asigna a la variable X el doble del valor de A, si el valor de la variable A + 3 es menor que el de la variable B al cuadrado y si el valor de la variable C al cuadrado es mayor que 3.

Observaciones

La condición que se escribe después de la palabra **IF** puede utilizar los siguientes símbolos de relación:

> , mayor que;	> = , mayor o igual a
< , menor que;	< = , menor o igual a
= , igual a;	< > , distinto a

y los operadores lógicos

AND: Y; OR: O; NOT: NO

STOP

Esta instrucción detiene la ejecución del programa en la línea en que se encuentra esta instrucción.

Ejemplo

La instrucción

```
100 STOP
```

hace que la ejecución del programa se detenga en la línea **100.**

CONT

Esta instrucción reanuda la ejecución del programa en la línea siguiente a la que ha sido interrumpido con la instrucción STOP.



Ejemplo

Si un programa contiene estas instrucciones

```
100 STOP
110 PRINT "CONTINUA"
```

la ejecución se detiene en la línea **100.** Cuando se pulsa **CONT** se ejecuta la instrucción siguiente **110,** y en la pantalla se imprime la palabra CONTINUA.

PROGRAMAS RESUELTOS EXPLICADOS

4.1. Escribe en un papel lo que escribiría el microordenador en la pantalla al ejecutarse este programa:

```
10 LET A = 0
20 PRINT "YA ESTA BIEN DE REPETICIONES!"
30 LET A = A + 1
40 PRINT A
50 PRINT
60 GOTO 20
70 END
```

```
¡YA ESTA BIEN DE REPETICIONES!
1
¡YA ESTA BIEN DE REPETICIONES!
2
.....
```

— Inicialmente la variable A vale 0 **(10 LET A = 0).** A continuación se escribe la frase por primera vez (instrucción **20**). La variable A toma a continuación el valor 1 **(30 A = A + 1),** y se imprime este valor **(40 PRINT A).** La instrucción **50 PRINT** hace saltar una línea.

— El proceso se repite incrementándose la variable A en una unidad cada vez que la instrucción **60 GOTO 20** transfiere el control a la instrucción **20**.

4.2. Indica qué instrucciones nunca se ejecutan en el siguiente programa:

```
10  LET A = 5.1
20  LET B = 8.9
30  LET S = A + B
40  GOTO 70
50  LET P = A * B
60  PRINT P
70  PRINT S
80  END
```

Como la instrucción **40 GOTO 70** transfiere el control a la instrucción **70,** no se ejecutan nunca las instrucciones **50** y **60.**

En las instrucciones **10** y **20,** A y B toman los valores 5.1 y 8.9, respectivamente. En la instrucción **30,** S toma el valor de su suma, 14.0, que se imprimirá en la pantalla mediante la instrucción **70 PRINT S.**

4.3. Elabora un programa que escriba un número ilimitado de veces la frase ESTOY PROGRAMANDO CON LA INSTRUCCION GOTO e imprime el número de veces que lo escribe.

```
10  LET A = 0
20  PRINT "ESTOY PROGRAMANDO CON LA INSTRUCCION GOTO"
```

```
30  LET A = A + 1
40  PRINT A
50  GOTO 20
60  END
```

Este programa, salvo la frase que se imprime, es idéntico al del ejercicio 4.1.

4.4. Haz un programa que escriba todos los números naturales.

```
 1  REM NUMEROS NATURALES
10  LET N = 0
20  PRINT N
30  LET N = N + 1
40  GOTO 20
50  END
```

— En la instrucción **10** se da a la variable N el valor 0 que se imprime en la instrucción **20.**
— En la instrucción **30** se añade una unidad a N que ahora vale 1.
— La instrucción **40** transfiere el control a la instrucción **20** que imprime de nuevo este número.
— En la instrucción **30** N se incrementa una unidad y vale ahora 2, se salta de nuevo a la instrucción **20** que imprime este número. El proceso continúa así indefinidamente apareciendo en la pantalla los sucesivos números naturales.

4.5. Escribe en un papel lo que aparecería en la pantalla si se ejecutara el siguiente programa:

```
10  LET C = 0
20  PRINT C
30  PRINT
40  LET C = C + 2
50  GOTO 20
60  END
```

```
0

2

4

6
```

Aparecen los números pares separados por espacios en blanco.

— En la instrucción **10** C toma el valor 0 y lo imprime (instrucción **20**). En la instrucción **30** se salta una línea.
— En la instrucción **40 LET C = C + 2** (C = 0 + 2) C toma el valor 2 y en la **50** se transfiere el control a la instrucción **20** que imprime este valor de C. El proceso sigue así indefinidamente.

4.6. Elabora un programa que imprima los sucesivos números impares en filas.

```
10 LET I = 1
20 PRINT I; " ";
30 LET I = I + 2
40 GOTO 20
50 END
```

— La variable I, que inicialmente vale 1, va imprimiéndose en la instrucción **20,** dejando un espacio en blanco entre cada número impar.
— En la instrucción **30** la variable I va incrementándose de 2 en 2 unidades.

4.7. Escribe en un papel lo que aparecerá en la pantalla al ejecutarse el siguiente programa, para N = 8.

```
10 INPUT N
20 LET A = 0
30 PRINT A; ",";
40 LET A = A + 1
50 IF A < N THEN GOTO 30
60 END
```

```
0, 1, 2, 3, 4, 5, 6, 7,
```

— En la instrucción **10** se introduce el número 8.
— A, inicialmente, vale 0 (instrucción **20**).
— En la instrucción **30** se imprimen los valores que toma A separados por comas.
— La instrucción **40 LET A = A + 1** incrementa en una unidad el valor de A.
— La instrucción **50** analiza si se ha alcanzado el valor introducido en la instrucción **10 INPUT N** (en este caso 8). Si este valor no se ha alcanzado se transfiere el control a la instrucción **30,** en caso contrario el programa se detiene.

4.8. Escribe un programa que calcule e imprima los múltiplos de 5 hasta un cierto número.

```
 1 REM MULTIPLOS DE 5
10 INPUT N
20 LET M = 5
30 PRINT M
40 LET M = M + 5
50 IF M < N THEN GOTO 30
60 END
```

— En la instrucción **10** se introduce el número N hasta el cual se calcularán los múltiplos de 5.
— En la instrucción **20 LET M = 5,** M toma como valor inicial 5 que se imprime en **30 PRINT M.**
— En la instrucción **40 LET M = M + 5,** M incrementa su valor en 5.
— La instrucción **50** analiza si se ha alcanzado el valor tope introducido en **10.** Si esto no ha ocurrido todavía se transfiere el control a **30** para que el proceso continúe. En caso contrario el programa termina.

4.9. Escribe un programa que calcule e imprima los números pares naturales hasta un cierto número.

```
10 INPUT N
20 LET M = 0
30 PRINT M
40 LET M = M + 2
50 IF M < N THEN GOTO 30
60 END
```

El programa es muy semejante al anterior, salvo que en la instrucción **20 LET M = 0,** M toma el valor inicial 0 y en la instrucción **40** su valor se incrementa de 2 en 2.

4.10. Indica qué hace el siguiente programa:

```
 10 INPUT N
 20 LET A = 0
 30 PRINT A; ", ";
 40 LET A = A + 7
 50 IF A < N THEN GOTO 30
 60 PRINT
 70 PRINT
 80 LET B = 0
 90 PRINT B; ", ";
100 LET B = B + 11
110 IF B < N THEN GOTO 90
120 END
```

— Imprime los múltiplos de 7, separados por comas, hasta un cierto número N.
— A continuación obtiene los múltiplos de 11 hasta ese mismo número (también separados por comas).
— Entre los múltiplos de 7 y los múltiplos de 11 las instrucciones **60** y **70** hacen que se salten dos líneas.

4.11. Haz un programa que calcule e imprima primero los múltiplos de 19 menores que 216 y después los de 23.

```
 10 LET N = 216
 20 LET A = 0
 30 PRINT A; ", ";
 40 LET A = A + 19
 50 IF A < N THEN GOTO 30
 60 PRINT
 70 PRINT
 80 LET B = 0
 90 PRINT B; ", ";
100 LET B = B + 23
110 IF B < N THEN GOTO 90
120 END
```

El programa es similar al 4.10. En la instrucción **10** se fija el tope numérico en 216. La instrucción **40 LET A = A + 19** hace que el valor de A se incremente de 19 en 19 unidades desde A igual a 19 hasta llegar al máximo de 19 menor de 216.

Después de dejar dos huecos en blanco el proceso se repite para los múltiplos de 23.

4.12. Elabora un programa que introduzca mediante **INPUT** una variable, de tal manera que, si ésta tiene un valor mayor que cero, el programa vuelve a pedir un nuevo valor y, en caso contrario, termine la ejecución.

```
10 INPUT X
20 IF X > 0 THEN GOTO 10
30 END
```

— La instrucción **10** permite introducir el valor de la variable X. Si el número es mayor que cero el control se transfiere a **10.** Si el valor es cero o negativo el programa termina.

4.13. Explica lo que hace este programa:

```
10 INPUT A
20 IF A < 10 AND A > 5 THEN GOTO 50
30 PRINT "NO SE CUMPLE"
40 GOTO 60
50 PRINT "SI SE CUMPLE"
60 END
```

Escribe la frase SI SE CUMPLE cuando se introduce un número que sea a la vez menor que 10 y mayor que 5, es decir, 6, 7, 8 y 9. En caso contrario, escribe la frase NO SE CUMPLE.

4.14. Escribe un programa que introduzca un número y que imprima el signo + si este número es positivo y el signo − si es negativo.

```
10 INPUT N
20 IF N > = 0 THEN PRINT "+" : GOTO 40
30 PRINT "−"
40 END
```

En la instrucción **10 INPUT N** se introduce el valor del número. Si N es cero o positivo se imprime el signo +. Si no se cumple la condición (N es negativo) se imprime el signo −.

4.15. Explica cada una de las instrucciones del programa siguiente:

```
10 LET C = −1
20 LET C = C + 1
30 PRINT C
40 PRINT "INTRODUCE EL PESO"
50 INPUT X
60 IF X > 50 AND X < 60 THEN GOTO 20
70 GOTO 40
80 END
```

— La instrucción **10** asigna a C el valor −1.
— La instrucción **20** añade a C una unidad (así C inicialmente vale 0).
— La instrucción **30** imprime C.
— La instrucción **40** imprime la frase "INTRODUCE EL PESO".
— La instrucción **50** pide un valor y lo asigna a la variable X.
— La instrucción **60** analiza si el valor de X es mayor que 50 y a la vez menor que 60. Si se cumple la condición se transfiere el control a la instrucción **20.**

En caso de no cumplirse, la instrucción **70** transfiere el control a la instrucción **40.**

El programa permite contar cuántos valores del peso son a la vez mayores de 50 y menores de 60.

4.16. Escribe un programa que calcule el número de alumnos cuyo peso sea mayor que 60 kilos y su estatura inferior a 1.65 metros.

```
 1  REM ALUMNOS DE PESO > 60 Y ALTURA < 1.65
10  LET S = −1
20  LET S = S + 1
30  PRINT "EL NUMERO TOTAL, HASTA EL MOMENTO ES : ", S
40  INPUT P
50  INPUT A
60  IF P > 60 AND A < 1.65 THEN GOTO 20
70  GOTO 30
```

— En las instrucciones **40** y **50** se introducen el peso y la altura. Si se cumple la condición establecida se transfiere el control a la instrucción **20** en la que se incrementa en una unidad el valor de S (que inicialmente es −1) y se imprime este valor en la pantalla. Si no se cumple la condición, se transfiere el control a la instrucción **30** directamente y se imprime el valor que ya tenía S.

4.17. Elaborar un programa que dé los múltiplos comunes de 3 y 7 hasta un cierto número.

Los múltiplos de 3 y 7 a la vez son 21, 42, 63, 84, es decir, los múltiplos de 21, así un programa que los calcula es

```
10  INPUT N
20  LET M = 21
30  PRINT M
40  LET M = M + 21
50  IF M < N THEN GOTO 30
60  END
```

4.1. ¿El siguiente programa imprimirá algo en la pantalla? Da una explicación.

```
10  LET R = 5 ↑ 3
20  GOTO 40
30  PRINT R
40  END
```

No imprime nada, puesto que salta a la instrucción **40** sin pasar por la instrucción **30 PRINT R** que no se ejecuta.

4.2. Escribe un programa que imprima, en columna, 8 veces el número 5.1.

```
10  LET N = 0
20  LET N = N + 1
30  IF N < = 8 THEN PRINT 5.1 : GOTO 20
40  END
```

4.3. Modifica el programa anterior para que escriba los números en fila.

Se modificaría la instrucción **30** así:

```
30  IF N <  = 8 THEN PRINT 5.1; ", "; : GOTO 20
```

4.4. Explica lo que hace el siguiente programa:

```
10  LET C = 0
20  INPUT "INTRODUCE UN NUMERO"; N
30  LET C = C + 1
40  IF C > N THEN GOTO 70
50  PRINT "SIGUE"
60  GOTO 30
70  PRINT "TERMINA"
80  END
```

Se pide un número en la instrucción **20.** Si la variable C es mayor que este número escribe en la pantalla la palabra TERMINA. Si es menor o igual imprime la frase SIGUE.

4.5. ¿Qué imprimirá el siguiente programa para X = 4? ¿Y para X = 83?

```
10  INPUT "INTRODUCE EL NUMERO X"; X
20  IF X < 50 THEN GOTO 40
30  STOP
40  PRINT X
50  GOTO 10
60  END
```

Para X = 4 imprime 4 en la pantalla y pide un nuevo número.
Para X = 83 se para sin imprimir nada.

4.6. Escribe en tu propio lenguaje las siguientes condiciones expresadas en lenguaje BASIC:

a) **IF A < 5 THEN GOTO 70**
b) **IF A < > −1 THEN GOTO 20**
c) **IF X <20 AND X >10 THEN GOTO 100**
d) **IF X < > Y OR X = Z THEN GOTO 80**

a) Si A es menor que 5 vete a la instrucción **70.**
b) Si A es distinto de −1 vete a la instrucción **20.**
c) Si X es menor que 20 y X mayor que 10 vete a la instrucción **100.**
d) Si X es distinto de Y o X es igual a Z ve a **80.**

4.7. Escribe las siguientes condiciones en lenguaje BASIC:

a) Si el valor de X es menor o igual que cero pasa a la instrucción **30.**
b) Si el valor de B_1 es mayor que el valor B_2 pasa a la instrucción **350.**
c) Si el valor de X es mayor que Y y mayor que Z pasa a la instrucción **500**.
d) Si el valor de X es mayor que Y o mayor que Z pasa a la instrucción **320.**

a) IF X < = 0 THEN GOTO 30
b) IF B1 > B2 THEN GOTO 350
c) IF X > Y AND X > Z THEN GOTO 500
d) IF X > Y OR X > Z THEN GOTO 320

4.8. Escribe en lenguaje BASIC:

a) Si Z es mayor o igual que A al cuadrado, entonces pasa a la línea **150.**
b) Si 3 veces A es igual a 12 entonces pasa a la línea **80.**

a) **IF Z > = A ↑ 2 THEN GOTO 150**
b) **IF 3 * A = 12 THEN GOTO 80**

4.9. Escribe en una frase lo que expresan las siguientes instrucciones:

a) **10 IF A > B + 2 THEN GOTO 500**
20 GOTO 200
b) **10 IF C > N THEN GOTO 205**
20 LET C = C + 1

a) **10** Si el valor de A es mayor que el de B más 2, entonces vete a la instrucción **500.**
20 Vete a la instrucción **200.**
b) **10** Si el valor de C es mayor que N, entonces vete a **205.**
20 Haz el valor de C igual al anterior valor más uno.

4.10. Escribe una o más instrucciones en lenguaje BASIC que expresen lo siguiente:

a) Si X es mayor que Y más 2, ir a la instrucción **220;** en caso contrario, ir a la instrucción **125.**

b) Si I es igual a cero, ir la instrucción **150,** y si no es así aumenta el valor de J en una unidad.

a)
```
10 IF X > Y + 2 THEN GOTO 220
20 GOTO 125
```
b)
```
10 IF I = 0 THEN GOTO 150
20 LET J = J + 1
```

4.11. Indica cuál de las dos instrucciones es incorrecta y explica por qué.

a) **5 IF X ↑ 2 = 100 THEN GOTO 50**
b) **30 IF X < = 50 THEN X = X + 5**

a) Esta instrucción es correcta.

b) En algunos microordenadores no es correcta, pues después del **THEN** aparece una instrucción de asignación sin **LET.**

4.12. ¿Qué imprimirá el siguiente programa al ser ejecutado, si se introduce 8, 7 y 5 para la variable X?

```
10 LET T = 0
20 INPUT X
30 LET T = T + 5 * X
40 PRINT T
50 GOTO 20
60 END
```

Para X = 8 imprime 40, a continuación para X = 7 imprime 75 **(T = = 40 + 5 * 7),** y después para X = 5 imprime 100 **(T = 75 + 5 * 5 = = 100).**

4.13. Añade una o más instrucciones para que la ejecución del programa anterior termine después de imprimirse T, para los mismos valores de X.

Para que la ejecución termine después de imprimir estos valores basta añadir las instrucciones

```
35 IF X = 5 THEN GOTO 50
50 END
```

4.14. Escribe lo que aparecería en la pantalla si se ejecutara el programa siguiente para X = 5, −3, 6, 0, 8, −1.

```
10 INPUT X
20 IF X < 0 THEN GOTO 10
30 PRINT X
40 GOTO 10
50 END
```

```
?  5
 5
?  –3
?  6
 6
?  0
 0
?  8
 8
?  –1
?
```

Imprime los números positivos y el cero en la pantalla, pero no los negativos.

4.15. El siguiente programa ordena los números A y B.

```
10 INPUT "INTRODUCE LOS DOS NUMEROS"; A, B
20 IF A < B THEN GOTO 50
30 IF A > B THEN GOTO 60
40 IF A = B THEN GOTO 70
50 PRINT A; "ES MENOR QUE"; B
60 PRINT A; "ES MAYOR QUE"; B
70 PRINT A; "ES IGUAL QUE"; B
80 GOTO 10
90 END
```

Sin embargo, al introducir, por ejemplo, los números 5 y 7 se obtiene en la pantalla los siguientes resultados:

```
5  ES MAYOR QUE 7
5  ES MAYOR QUE 7
5  ES IGUAL A 7
```

Modifica el programa para corregir este fallo.

Introduciendo instrucciones **GOTO 10** después de cada **PRINT** para que sólo se ejecute uno de ellos cada vez.

Por tanto, basta añadir las instrucciones

```
55 GOTO 10
65 GOTO 10
```

4.16. Escribe un programa que lea A y B y compare ambos valores de tal manera que si A > B entonces calcule e imprima A − B, y, en caso contrario, calcule e imprima B − A.

```
10  INPUT A
20  INPUT B
30  IF A > B THEN GOTO 60
40  PRINT B − A
50  GOTO 70
60  PRINT A − B
70  END
```

Otra forma de escribir el programa es

```
10  INPUT A, B
20  IF A > B THEN PRINT A − B : GOTO 40
30  PRINT B − A
40  END
```

4.17. Escribe un programa que lea dos números y los compare de tal manera que si el primero es mayor que el segundo, escriba su diferencia y su cociente; y en caso contrario, escriba su suma y su producto.

```
10  INPUT A
20  INPUT B
30  IF A > B THEN GOTO 70
40  PRINT "LA SUMA ES:", A + B
50  PRINT "EL PRODUCTO ES : ", A * B
60  GOTO 90
70  PRINT "LA DIFERENCIA ES : ", A-B
80  PRINT "EL COCIENTE ES:", A/B
90  END
```

Un programa que realiza la misma función es

```
10  INPUT A, B
20  IF A > B THEN PRINT A − B, A/B : GOTO 40
30  PRINT A + B, A * B
40  END
```

4.18. Para conocer la opinión del público sobre un determinado producto se hace una encuesta, preguntando a un conjunto de personas sobre dicho producto. Las respuestas favorables se indican con 1 y las desfavorables, con 0. Elabora un programa que cuente las preguntas favorables y desfavorables.

Al introducir −1 se da por finalizado el proceso

```
 10  LET F = 0
 20  LET D = 0
 30  INPUT X
 40  IF X = −1 THEN GOTO 100
 50  IF X = 0 THEN GOTO 80
 60  LET F = F + 1
 70  GOTO 30
 80  LET D = D + 1
 90  GOTO 30
100  PRINT "RESPUESTAS FAVORABLES", F
110  PRINT "RESPUESTAS DESFAVORABLES", D
120  END
```

4.19. Escribe un programa que calcule la velocidad de un movimiento uniformemente variado, si la aceleración *a* es positiva. (La fórmula de la velocidad en este movimiento es $v = v_0 + a\ t$, donde v_0 es la velocidad inicial, a la aceleración, t el tiempo y v la velocidad final.)

```
10  INPUT V0
20  INPUT A
30  INPUT T
40  IF A > 0 AND T > = 0 THEN GOTO 60
50  GOTO 10
60  PRINT "LA VELOCIDAD ES : "; V0 + A * T; "M/S"
70  END
```

4.20. Haz un programa que calcule la media de una serie de números ordenados en forma creciente, siendo el último número de la serie el 1,523.

```
10  LET S = 0 : LET N = 0
20  INPUT A
30  LET S = S + A
40  LET N = N + 1
50  IF A = 1.523 THEN PRINT "LA MEDIA ES : "; S/N : GOTO 70
60  GOTO 20
70  END
```

4.21. Explica lo que hace el siguiente programa:

```
10  INPUT "NOTA" : X
20  IF X < 5 THEN GOTO 40
30  IF X > = 5 THEN GOTO 50
40  PRINT "NO APROBADO"
50  PRINT "APROBADO"
60  GOTO 10
70  END
```

Cuando se introduce una nota numérica si es mayor o igual que 5 se imprime APROBADO.

Si la nota es menor que 5 se imprime NO APROBADO y a continuación APROBADO.

Para evitar que se imprima esta segunda frase basta añadir la instrucción **45 GOTO 10.**

4.22. Escribe un programa que lea el salario mensual de un conjunto de personas, y cuando encuentre un salario superior a 200.000 pesetas, pare la ejecución, pero de tal manera que permita continuarla cuando se desee.

```
10  INPUT "INTRODUZCA EL SALARIO MENSUAL "; S
20  IF S > 200000 GOTO 40
30  GOTO 10
40  END
```

4.23. Explica cómo funciona el siguiente programa y qué es lo que hace.

```
10  LET X = 0
20  LET X = X + 10
30  IF X > 100 THEN GOTO 60
40  PRINT X
50  GOTO 20
60  END
```

— Escribe los múltiplos de 10 hasta 100.
— En la instrucción **10** se asigna la variable X el valor 0. En la instrucción **20** X se aumenta en 10 unidades.
— En la instrucción **30** si el valor de X es superior a 100 se detiene la ejecución del programa. Si es menor que 100 se imprime el valor de X y se transfiere después el control a la instrucción **20.**

4.24. Escribe un programa similar al anterior que imprima los múltiplos de 5 comprendidos entre 1535 y 1640, excluidos estos números.

```
 5  REM MULTIPLOS DE 5 COMPRENDIDOS ENTRE 1535 y 1640
10  LET X = 1535
20  LET X = X + 5
30  IF X = 1640 THEN GOTO 60
40  PRINT X
50  GOTO 20
60  END
```

4.25. Hacer un programa que averigüe si un número es par o impar.

```
 1 REM NUMERO PAR O IMPAR
10 INPUT X
20 LET N = 0
30 LET N = N + 1
40 IF X = 2 * N THEN GOTO 80
50 IF X > 2 * N THEN GOTO 30
60 PRINT X, "ES IMPAR"
70 GOTO 90
80 PRINT X, "ES PAR"
90 END
```

4.26. Explica cómo funciona el siguiente programa y lo que hace.

```
10 LET A = 32
20 LET A = A + 3
30 IF A < 51 AND A > 29 THEN GOTO 50
40 STOP
50 PRINT A
60 GOTO 20
70 END
```

— Imprime los números 35, 38, 41, 44, 47 y 50.
— En la instrucción **10 LET A = 32,** A toma el valor 32.
— La instrucción **20** aumenta en tres unidades el valor de la variable A.
— Si A es menor que 51 y A mayor que 29 (lo que siempre ocurre ya que inicialmente A vale 32) se transfiere el control a la instrucción **50** que imprime el valor de A.
— La instrucción **60** transfiere de nuevo el control a **20** que inicia de nuevo el proceso.

4.27. Escribe un programa que introduzca números sucesivamente, averigüe cuál es el mayor de los introducidos hasta el momento y dé el informe correspondiente.

```
 1 REM MAYOR HASTA EL MOMENTO
10 INPUT X
20 PRINT "EL MAYOR ES", X
30 INPUT Y
40 IF Y < = X THEN GOTO 20
50 LET X = Y
60 GOTO 20
70 END
```

4.28. Modifica el programa anterior de tal manera que escriba en una columna el número introducido y en otra, el mayor introducido hasta el momento. Como cabeceras de ambas columnas deberá escribir

NUMERO   EL MAYOR

```
 1 REM EL MAYOR HASTA EL MOMENTO
10 PRINT "NUMERO", "EL MAYOR"
20 INPUT Y
30 LET X = Y
40 PRINT Y, X
50 INPUT Y
60 IF Y < = X THEN GOTO 40
70 GOTO 30
80 END
```

4.29. Dados cuatro números A, B, C y D, escribir un programa que haga I = 1 si los cuatro son iguales; I = 2 si de los cuatro, tres son iguales; I = 3 si dos son iguales; I = 4 si son iguales dos a dos; e I = 5 si todos son diferentes.

```
 1 REM CLASIFICADOR DE NUMEROS
10 INPUT A, B, C, D
20 IF A = B AND A = C AND A = D THEN LET I = 1 : PRINT I
30 IF (A = B AND (A = C OR A = D)) OR (C = D AND (C = A OR
   C = B)) THEN LET I = 2 : PRINT I
40 IF A = B OR A = C OR A = D OR B = C THEN LET I = 3 :
   PRINT I
50 IF (A = B AND C = D) OR (A = C AND B = D) OR (A = D
   AND B = C) THEN LET I = 4 : PRINT I
60 IF A < > AND A < > C AND A < > D THEN I = 5 : PRINT
70 END
```

5 Datos de un programa

Programas resueltos con READ, DATA y RESTORE

RESUMEN DE LAS INSTRUCCIONES

READ-DATA

La instrucción READ lee y asigna a variables los datos contenidos en una o más instrucciones DATA.



Ejemplos

- Al ejecutarse las instrucciones

```
10  DATA 22, 30, 70.8
20  READ A, B
30  READ C
```

  los datos contenidos en la instrucción **10** se asignan a las variables numéricas A, B y C, respectivamente.

- Al ejecutarse las instrucciones

```
 10  READ A, B
 20  READ C
100  DATA 8.9, 3
110  DATA 5
```

  los datos contenidos en la instrucción **100** (8.9 y 3) se asignan a las variables A y B, respectivamente. El dato contenido en la instrucción **110,** el número 5, se asigna a la variable C.

Observaciones

- La posición de la instrucción **READ** en un programa es importante, en cambio, las instrucciones **DATA** pueden colocarse en cualquier lugar del mismo. Cuando el microordenador encuentra una instrucción **READ** lee el dato siguiente al último que se leyó anteriormente, independientemente de la posición que ocupen las instrucciones **DATA** en el programa.

- Si se intentan leer más datos que los contenidos en las instrucciones **DATA** se produce un mensaje de error.

RESTORE

Esta instrucción permite volver a leer un conjunto de datos.



Ejemplo

Al ejecutarse las instrucciones

```
10 DATA 22, 30, 70.8
20 READ A, B
30 READ C
40 RESTORE
50 READ D, E
60 READ F
```

se asignan los valores 22, 30 y 70.8 a las variables A, B y C en primer lugar, y después a las variables D, E y F.

Si no estuviera la instrucción **40** aparecería en la pantalla un mensaje de error en la instrucción **50,** al intentar leer datos y no disponer de ellos en instrucciones DATA.

PROGRAMAS RESUELTOS EXPLICADOS

5.1. ¿Qué datos no se leerán al ejecutar el siguiente programa?

```
10 READ A, B, C
20 PRINT B
30 READ X, Y
40 PRINT A, Y
50 READ U, V, W, L
60 DATA 1.1, 1.2, 1.3, 1.4
70 DATA 2.3, 3.3, 3.7, 4.8, 5.9, 7.3
80 DATA −2.3, −4.5
90 END
```

Puesto que en las dos instrucciones en las que se manda leer **(30 READ X, Y y 50 READ U, V, W, L)** hay cinco variables, se leen los cinco primeros datos y quedan sin leer 3.3, 3.7, 4.8, 5.9, 7.3, −2.3, −4.5.

5.2. Elabora un programa que compare un número con la siguiente serie de datos: 1, 3, 5, 7 y 9, dé un informe afirmativo en el caso de que el número introducido coincida con alguno de la serie y dé un mensaje de error, a la falta de datos, en caso de que no coincida.

```
 10 LET N = 0
 20 INPUT X
 30 READ Y
 40 IF X = Y THEN PRINT "COINCIDE" : GOTO 80
 50 LET N = N + 1
 60 IF N < 5 THEN GOTO 30
 70 PRINT "NO COINCIDE"
 80 END
100 DATA 1, 3, 5, 7, 9
```

— En la instrucción **40** el número que se ha asignado a X en la instrucción **20** se compara con el leído en la instrucción **30,** escribiendo el mensaje COINCIDE cuando así ocurre, transfiriendo a continuación el control al fin del programa.
— En la instrucción **50** la variable aumenta en una unidad.
— La instrucción **60** hace que se repita el proceso siempre que el valor de N sea menor que 5 (número de datos disponible).

5.3. Escribe un programa que contenga una instrucción **DATA** con los diez primeros números enteros positivos y −1 en último lugar. Dicho programa deberá ir sumando los diez primeros números y cuando lea −1 imprimirá la suma; a continuación que detenga la ejecución.

```
 10 LET S = 0
 20 READ A
 30 IF A = -1 THEN GOTO 60
 40 LET S = S + A
 50 GOTO 20
 60 PRINT S
 70 END
100 DATA 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, -1
```

En la variable S (definida en la instrucción **10**) se van acumulando los distintos números leídos (instrucción **40**). Cuando el número leído es −1, se imprime la suma.

5.4. ¿Por qué razón en el siguiente programa las instrucciones **READ** se comienzan a leer siempre por el primer dato, sin seguir el orden secuencial?

```
10 READ A
20 RESTORE
30 READ H, I, J
```

```
40 RESTORE
50 READ X, Y
60 DATA 2, −3, 4, −5, 2.3, 7
70 END
```

Se debe a la presencia de la instrucción **RESTORE** en las instrucciones **20** y **40** que desplazan un hipotético indicador de lectura hacia el primer dato contenido en la instrucción **DATA.**

5.5. Añade instrucciones **RESTORE** al siguiente programa para que no se interrumpa su ejecución por falta de datos.

```
10 READ X, Y
20 PRINT X, Y
30 READ A, B, C
40 PRINT (A * B)/C
50 READ L, M
60 PRINT L ↑ M
70 DATA 4, 7, 2
80 END
```

```
10 READ X, Y
20 PRINT X, Y
25 RESTORE
30 READ A, B, C
40 PRINT (A * B)/C
45 RESTORE
50 READ L, M
60 PRINT L ↑ M
70 DATA 4, 7, 2
80 END
```

Las instrucciones **25** y **45** permiten que las instrucciones **READ** tengan datos suficientes ya que trasladan la lectura al primer dato contenido en la instrucción **70.**

PROGRAMAS RESUELTOS

5.1. Las siguientes instrucciones contienen errores. Identifícalos.

a) 30 READ X, 30
b) 10 READ X, Y; Z
c) 40 DATA 5, 6, 7.5, X, 3.4
d) 70 DATA 3, 4; 5

a) La instrucción es incorrecta por contener el número 30, que no es una variable.
b) La instrucción es incorrecta porque entre las variables Y y Z hay un punto y coma en lugar de una coma.
c) Es incorrecta, pues una variable como X no puede estar incluida en una instrucción **DATA**.
d) Es incorrecta, puesto que los datos deben estar separados por comas, y entre 4 y 5 hay un punto y coma.

5.2. Un programa contiene las siguientes instrucciones:

```
 30 READ A, B, C
.............................................
 60 RESTORE
 70 READ W, X, Y, Z
.............................................
200 DATA 1, 3, 5, 7, 9, 11, 13
```

¿Qué valores se asignan a cada variable? ¿Qué valores se les asignarían si no estuviese la instrucción **60 RESTORE**?

Los valores que se asignan son:

A = 1
B = 3
C = 5

Y como aparece la instrucción **RESTORE** se vuelven a asignar datos desde el principio, según se indica:

W = 1
X = 3
Y = 5
Z = 7

Si se suprime la instrucción **60 RESTORE** la asignación es ésta:

A = 1
B = 3
C = 5
W = 7
X = 9
Y = 11
Z = 13

5.3. Escribe las instrucciones **READ** y **DATA** que asignan los valores −1.6 E−6, −500, 0.4077, 100, 110 a las variables C1, C2, C3, C4, C5.

```
10 READ C1, C2, C3, C4, C5
20 DATA −1.6 E −6, −500, 0.4077, 100, 110
```

5.4. ¿Qué imprimiría en la pantalla el microordenador al ejecutar este programa?

```
10 READ A, B
20 PRINT A, B
30 READ C
40 PRINT A + B + C
50 DATA 2, 7
60 END
```

```
2        7
OUT OF DATA IN 30
```

y al intentar leer la variable C, daría un error por falta de datos.

5.5. ¿Qué valores imprimirá el siguiente programa al ejecutarse?

```
10 READ X
20 PRINT X
30 IF X < > 3 THEN GOTO 10
40 DATA 7, 6, 5, 4, 3, 2, 1
50 END
```

Imprimirá los números 7, 6, 5, 4 y al leer 3, se detendrá el programa.

5.6. Diseña un programa que contenga en una instrucción **DATA** los números primos 2, 3, 5, 7, 11, 13, 19 y 23, y que al introducir un número positivo menor que 25 imprima si es primo o no.

```
10 LET N = 0
20 INPUT "INTRODUCE UN NUMERO"; A
30 IF A < 0 OR A > 25 THEN GOTO 20
40 READ X
50 IF A = X THEN PRINT "ES PRIMO" : GOTO 90
```

```
 60  LET N = N + 1
 70  IF N < = 9 THEN GOTO 40
 80  PRINT "NO ES PRIMO"
 90  END
100  DATA 2, 3, 5, 7, 11, 13, 17, 18, 23
```

5.7. Haz un programa que al introducir una cantidad en pesetas dé su equivalente en libras esterlinas, dólares, marcos, liras y francos.

El programa almacenará en una instrucción **DATA** las siguientes equivalencias:

1 libra = 210 ptas.
1 dólar = 170 ptas.
1 marco = 57 ptas.
1 lira = 0.09 ptas.
1 franco = 18 ptas.

```
  1  REM CAMBIO
 10  INPUT "INTRODUCE UNA CANTIDAD EN PESETAS "; P
 20  READ A, B, C, D, E
 30  PRINT P/A; " "; "LIBRAS"
 40  PRINT P/S; " "; "DOLARES"
 50  PRINT P/C; " "; "MARCOS"
 60  PRINT P/D; " "; "LIRAS"
 70  PRINT P/E; " "; "FRANCOS"
 80  END
100  DATA 210, 170, 57, 0.09, 18
```

5.8. En un observatorio se han registrado en una semana las siguientes temperaturas:

D	L	M	X	J	V	S
20	22	25	24	23	21	20

Escribe un programa que introduzca estos datos mediante instrucciones **READ** y **DATA**, y calcule la temperatura media.

```
  1  REM MEDIA DE TEMPERATURAS
 10  LET I = 0
 20  LET S = 0
 30  READ X
 40  LET I = I + 1
 50  LET S = S + X
 60  IF I < 7 THEN GOTO 30
 70  LET S = S/I
 80  PRINT "LA MEDIA ES:", S
 90  DATA 20, 22, 25, 24, 23, 21, 20
100  END
```

5.9. En una encuesta, las respuestas afirmativas se registran con 1 y las negativas con 2. Se encuesta a 50 personas, estando la última respuesta seguida del valor −1. Escribe un programa que incluya las siguientes instrucciones **DATA** y que cuente el número de respuestas afirmativas y negativas.

```
900 DATA 1, 2, 2, 1, 2, 2, 1, 1, 2, 1
910 DATA 2, 1, 1, 1, 2, 1, 2, 2, 2, 1
920 DATA 1, 2, 1, 2, 1, 1, 2, 1, 2, 2
930 DATA 1, 1, 1, 2, 2, 1, 1, 2, 2, 2
940 DATA 1, 2, 2, 2, 1, 1, 2, 1, 1, 1, −1
```

```
  1 REM ENCUESTA
 10 LET S = 0
 20 LET N = 0
 30 READ X
 40 IF X = − 1 THEN GOTO 100
 50 IF X = 1 THEN GOTO 80
 60 LET N = N + 1
 70 GOTO 30
 80 LET S = S + 1
 90 GOTO 30
100 PRINT "RESPUESTAS AFIRMATIVAS", S
110 PRINT "RESPUESTAS NEGATIVAS", N
900 DATA 1, 2, 2, 1, 2, 2, 1, 1, 2, 1
910 DATA 2, 1, 1, 1, 2, 1, 2, 2, 2, 1
920 DATA 1, 2, 1, 2, 1, 1, 2, 1, 2, 2
930 DATA 1, 1, 1, 2, 2, 1, 1, 2, 2, 2
940 DATA 1, 2, 2, 2, 1, 1, 2, 1, 1, 1, −1
950 END
```

5.10. El siguiente programa muestra cómo se pueden hacer dibujos sencillos con las instrucciones BASIC conocidas hasta ahora.

```
  1 REM IMPRIME UN DIBUJO
 10 READ X
 20 IF X = −1 THEN GOTO 920
 30 IF X = 1 THEN GOTO 60
 40 IF X = 2 THEN GOTO 80
 50 IF X = 3 THEN GOTO 100
 60 PRINT "***************"
 70 GOTO 10
 80 PRINT "***"
 90 GOTO 10
100 PRINT "**********"
110 GOTO 10
```

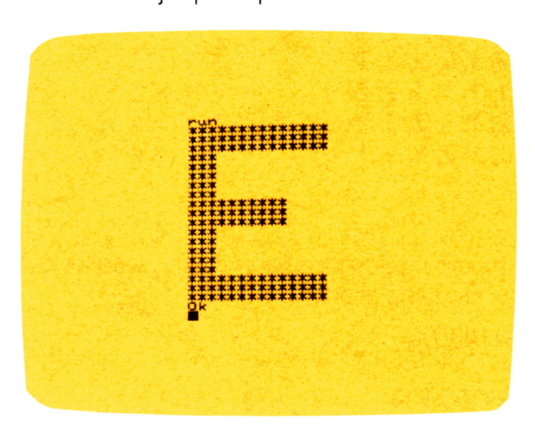
```
900  DATA 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3
910  DATA 2, 2, 2, 2, 2, 1, 1, 1, -1
920  END
```

¿Qué letra dibuja este programa?

Dado el programa:

```
  1  REM IMPRIME UN DIBUJO
 10  READ X
 20  IF X = -1 THEN GOTO 920
 30  IF X = 1 THEN GOTO 60
 40  IF X = 2 THEN GOTO 80
 50  IF X = 3 THEN GOTO 100
 60  PRINT "***************"
 70  GOTO 10
 80  PRINT "***"
 90  GOTO 10
100  PRINT "**********"
110  GOTO 10
900  DATA 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3
910  DATA 2, 2, 2, 2, 2, 1, 1, 1, -1
920  END
```

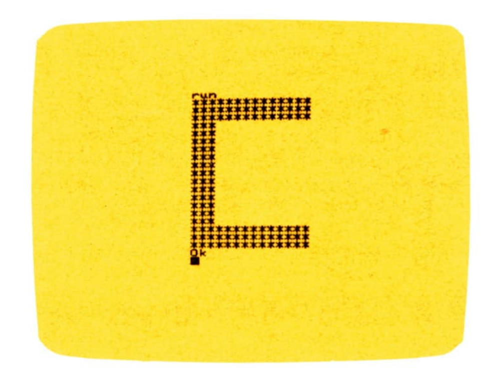
El dibujo que imprime es la letra E.



5.11. Basándote en el ejercicio anterior, haz un programa que dibuje la letra C en la pantalla.

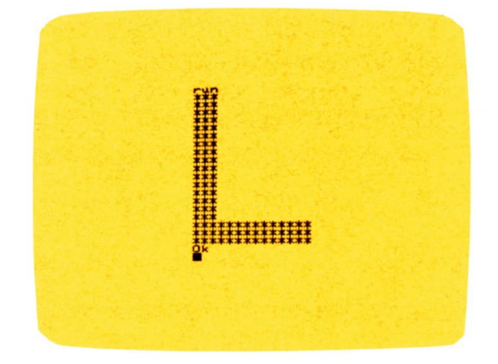
```
  1  REM IMPRIME LA LETRA C
 10  READ X
 20  IF X = -1 THEN GOTO 920
 30  IF X = 1 THEN GOTO 50
 40  IF X = 2 THEN GOTO 70
 50  PRINT "***************"
```

```
60  GOTO 10
70  PRINT "***"
80  GOTO 10
900 DATA 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2
910 DATA 2, 2, 2, 2, 2, 1, 1, 1, -1
920 END
```



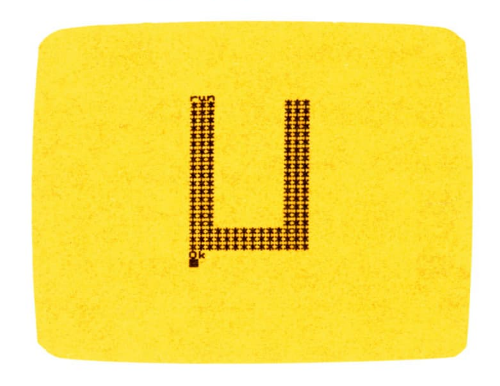
5.12. Diseña un programa que dibuje una L en la pantalla.

```
1   REM IMPRIME LA LETRA L
10  READ X
20  IF X = -1 THEN GOTO 920
30  IF X = 1 THEN GOTO 50
40  IF X = 2 THEN GOTO 70
50  PRINT "***************"
60  GOTO 10
70  PRINT "***"
80  GOTO 10
900 DATA 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2
910 DATA 2, 2, 2, 2, 2, 1, 1, 1, -1
920 END
```



5.13. Haz un programa que dibuje la letra U en la pantalla.

```
1   REM IMPRIME LA LETRA U
10  READ X
20  IF X = -1 THEN GOTO 920
30  IF X = 1 THEN GOTO 50
40  IF X = 2 THEN GOTO 70
50  PRINT "***************"
60  GOTO 10
70  PRINT "***         ***"
          (espacios en blanco)
80  GOTO 10
900 DATA 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2
910 DATA 2, 2, 2, 2, 2, 1, 1, 1, -1
920 END
```



6 Repetición de procesos: bucles

Programas resueltos con FOR-NEXT

RESUMEN DE LAS INSTRUCCIONES

FOR-NEXT

Estas instrucciones asociadas permiten repetir varias veces un proceso (bucle).

La primera instrucción bucle es FOR, y la última, NEXT.



Ejemplos

- Al ejecutar las instrucciones

```
10  FOR I = 1 TO 30
20  PRINT "PEPE"
30  NEXT I
```

la variable I toma inicialmente el valor 1 e imprime por primera vez la palabra PEPE. Continúa hasta que alcanza la instrucción **30 NEXT I,** y como no se ha alcanzado el tope establecido (en este caso 30), vuelve a la línea **20.** Este proceso se repite treinta veces tomando la variable I los valores desde 1 a 30.

- Al ejecutarse el programa

```
10  FOR I = 1 TO 30
20  PRINT I
30  NEXT I
```

los sucesivos valores que va tomando la variable I se imprimen en la pantalla del microordenador.

- Al ejecutarse las instrucciones

```
10 LET F = 4
20 LET A = 2
30 FOR I = 3 * A TO F ↑ 2
40 PRINT I
50 NEXT I
```

los valores de las variables F y A asignados en las instrucciones **10** y **20** se colocan en el bucle. De este modo, el programa imprime en la pantalla los valores de I que van desde 3 * 2 = 6 a 4 ↑ 2 = 16.

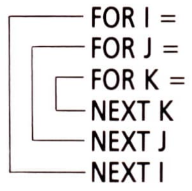
- Al ejecutarse el programa

```
10 FOR I = 1 TO 4
20 PRINT I
30 FOR J = 1 TO 6
40 PRINT J
50 NEXT J
60 NEXT I
```

que consta de un bucle incluido en otro (bucles anidados), en la pantalla se imprimen para cada valor de I (desde 1 a 4) los valores que toma la variable J (de 1 a 6).

Observaciones

- La estructura de los bucles anidados debe ser ésta:



No está permitida la estructura de bucles cruzados:

```
FOR I =
FOR J =
NEXT I
NEXT J
```

- No se puede entrar en un bucle sin pasar por la instrucción **FOR**. Así, este programa no es correcto.

```
10 FOR I = 1 TO 20
20 PRINT I
30 NEXT I
40 GOTO 20
50 END
```

Sin embargo, sí se puede salir de él en cualquier momento. Así, este otro programa sí es correcto.

```
10 FOR I = 1 TO 15
20 IF I = 3 THEN GOTO 40
30 NEXT I
40 PRINT I
50 END
```

STEP

La palabra STEP añadida a la instrucción FOR permite incrementar la variable con valores distintos de 1.



Ejemplos

- Al ejecutar la instrucción

  ```
  10 FOR I = 0 TO 30 STEP 2
  ```

  el valor de I varía desde 0 hasta 30 en pasos de 2. Es decir, adquiere sucesivamente los valores 0, 2, 4, 6 ... 30.

- Al ejecutar la instrucción

  ```
  10 FOR I = 20 TO O STEP −2
  ```

  la variable I toma valores de 20 a 0 en pasos de −2; es decir, adquiere sucesivamente los valores 20, 18, 16, ..., 0.

- Al ejecutar la instrucción

  ```
  10 FOR I = 1 TO 3 STEP 0.5
  ```

la variable toma valores de 1 a 3 en pasos de 0.5; es decir, adquiere sucesivamente los valores 1, 1.5, 2, 2.5, 3.

PROGRAMAS RESUELTOS EXPLICADOS

6.1. Indica qué escribiría en la pantalla el siguiente programa:

```
10 FOR N = 1 TO 6
20 PRINT "LA RAIZ CUADRADA DE"; N; "ES"; SQR(N)
30 NEXT N
40 END
```

```
LA RAIZ CUADRADA DE 1 ES 1
LA RAIZ CUADRADA DE 2 ES 1.414214
LA RAIZ CUADRADA DE 3 ES 1.732050
LA RAIZ CUADRADA DE 4 ES 2
LA RAIZ CUADRADA DE 5 ES 2.236068
LA RAIZ CUADRADA DE 6 ES 2.449490
```

La variable N toma valores de 1 hasta 6 y para cada uno de ellos se calcula y se imprime la raíz cuadrada.

6.2. Escribe un programa que imprima en la pantalla

```
 1. VEO LA VIDA CON OPTIMISMO
 2. VEO LA VIDA CON OPTIMISMO
.........................................................
.........................................................
15. VEO LA VIDA CON OPTIMISMO
```

```
10 FOR I = 1 TO 15
20 PRINT I; "."; " "; "VEO LA VIDA CON OPTIMISMO"
30 NEXT I
```

Para cada valor de I se imprime el valor de esta variable seguido de un punto, y de un espacio. A continuación se escribe la frase que está entre comillas.

6.3. Haz un programa que confeccione una tabla de las raíces cuadradas de los números comprendidos entre 100 y 112, ambos inclusive, con la siguiente cabecera:

```
NUMERO          RAIZ CUADRADA
```

```
10 PRINT "NUMERO", "RAIZ CUADRADA"
20 FOR I = 100 TO 112
30 PRINT I, SQR(I)
40 NEXT I
50 END
```

— La instrucción **10** imprime las dos frases que hacen de cabecera en dos zonas de la pantalla ya que están separadas por una coma.
— Para cada valor que toma la variable I del bucle se imprime dicho valor y su raíz cuadrada, cada una en una zona de la pantalla.

6.4. Elabora un programa que imprima los 25 primeros números pares.

```
10  FOR I = 0 TO 50 STEP 2
20  PRINT I,
30  NEXT I
40  END
```

Los 25 primeros números naturales pares se consiguen imprimiendo la variable I que controla al bucle, y que varía entre 0 y 50 en pasos de dos unidades.

6.5. Indica qué imprime en la pantalla el siguiente programa:

```
10  FOR L = 12 TO 30 STEP 3
20  PRINT L;
30  NEXT L
40  END
```

Imprime en la pantalla los distintos valores que toma I pero juntos, ya que la instrucción **20 PRINT L**; termina con un punto y coma.

```
run

12  5  18  21  24  27  30
```

6.6. Escribe un programa que imprima en la pantalla los múltiplos de 5 comprendidos entre 100 y 150.

```
10  FOR I = 20 TO 30
20  PRINT 5 * I
30  NEXT I
40  END
```

La variable I toma valores entre 20 y 50, que multiplicados por 5 dan los múltiplos de este número entre 100 (5 × 20) y 150 (5 × 30).

6.7. Elabora un programa que escriba en la pantalla los treinta primeros números impares negativos.

```
10 FOR I = −1 TO −61 STEP −2
20 PRINT I
30 NEXT I
40 END
```

La variable I va tomando los valores −1, −3, −5, etc., hasta llegar a −61, que es el número impar negativo que ocupa el lugar 30.

6.8. Escribe los resultados que imprimiría el siguiente programa si fuera ejecutado.

```
10 LET A = −4
20 LET B = 5
30 FOR R = A + B TO A * B STEP −2
40 PRINT R, R ↑ 3
50 NEXT R
60 END
```

Como A = −4 y B = 5, la variable R puede variar desde A + B = = (−4 + 5) = 1 hasta A × B (4 * (−5)) = − 20, en pasos de −2.

Así, en la pantalla aparecerán los números desde 1 a −19 y sus correspondientes cubos, ocupando ambos números dos zonas distintas de la pantalla, ya que entre R y R ↑ 3 hay una coma.

```
 1      1
−1     −1
−3     −27
−5     −125
−7     −343
−9     −729
−11    −1331
−13    −2197
−15    −3375
−17    −4913
−19    −6859
```

6.9. Escribe lo que imprimiría el siguiente programa al ser ejecutado:

```
10 FOR J = 1 TO 4
20 FOR K = 1 TO 2
30 PRINT J; ","; K,
40 NEXT K
50 NEXT J
60 END
```

Observación: Advertir el diferente significado de las dos comas en la instrucción **30**: la coma entre comillas se imprime a continuación del valor de J y la coma no entrecomillada significa que el siguiente valor del par J, K se imprime en la segunda zona de la pantalla.

```
1,1        1,2
2,1        2,2
3,1        3,2
4,1        4,2
```

Para cada valor de la variable J (que varía entre 1 y 4), la variable K toma los valores 1 y 2. Cada valor de I y los correspondientes de J se imprimen separados por comas en zonas distintas de la pantalla.

6.10. Haz un programa que escriba en la pantalla los siguientes pares de valores y en la disposición que se indica.

```
5,1
5,2
5,3
6,1
6,2
6,3
```

```
10 FOR I = 5 TO 6
20 FOR J = 1 TO 3
30 PRINT I; ","; J
40 NEXT J
50 NEXT I
60 END
```

Al no existir una coma detrás de la J, en la instrucción **30** (como en el programa anterior), la impresión de cada valor de I acompañado de cada valor de J se hace en líneas distintas.

6.11. Elabora un programa que imprima la siguiente tabla. (Suponer una pantalla de dos zonas.)

```
1,1        1,2
2,1        2,2
3,1        3,2
4,1        4,2
5,1        5,2
```

```
10 FOR J = 1 TO 5
20 FOR K = 1 TO 2
30 PRINT J; ","; K,
40 NEXT K
50 NEXT J
60 END
```

El programa es muy semejante al 6.9, salvo que la variable J puede tomar más valores, hasta 5 inclusive.

6.12. Escribe un programa que genere el producto cartesiano de los conjuntos:

A = (1, 2, ..., 10)
B = (1, 2, ..., 5)

Es decir, el programa deberá generar los pares

1,1 1,2 1,3 1,4 1,5
2,1 2,2 2,3 2,4 2,5
....
....
....
10,1 10,2 10,3 10,4 10,5

```
10 FOR I = 1 TO 10
20 FOR J = 1 TO 5
30 PRINT I; ","; J; "  ";
40 NEXT J
50 PRINT
60 NEXT I
70 END
```

En el bucle interior, la variable J varía de 1 a 5, para cada valor de I (de 1 a 10).

Después de cada valor de J se deja un espacio en blanco.

La instrucción **50 PRINT** hace que después de cada valor de I se salte una línea.

6.13. Escribe un programa que imprima los resultados posibles que se obtienen al lanzar dos dados. (Algunos resultados posibles son: 1,1; 1,6; 2,3; 5,6; 6,6; etcétera.)

```
10 FOR I = 1 TO 6
20 FOR J = 1 TO 6
30 PRINT I; ","; J,
40 NEXT J
50 NEXT I
60 END
```

Este programa es muy semejante al anterior. Genera el producto cartesiano de los conjuntos A = {1, 2, 3, 4, 5, 6} y B = {1, 2, 3, 4, 5, 6}

6.14. Elabora un programa que genere las tablas de multiplicar del 1, 2, ..., y 10.

```
10 FOR I = 1 TO 10
20 FOR J = 1 TO 10
30 PRINT I; "*"; J; "="; I * J
40 NEXT J
50 NEXT I
60 END
```

En la pantalla van apareciendo los resultados.

```
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
.....
.....
.....
.....
9 * 7 = 63
9 * 8 = 72
9 * 8 = 81
.....
.....
```

PROGRAMAS RESUELTOS

6.1. Analiza los dos programas siguientes. ¿Qué imprimen en la pantalla? Obtén una conclusión.

```
10 LET F = 1
20 IF F > 8 THEN GOTO 99
30 PRINT "F="; F
40 LET F = F + 1
50 GOTO 20
99 END
```

```
10 FOR F = 1 TO 8
20 PRINT "F="; F
30 NEXT F
99 END
```

El primero imprimirá:

```
F = 1
F = 2
F = 3
F = 4
F = 5
F = 6
F = 7
F = 8
```

El segundo imprimirá:

```
F = 1
F = 2
F = 3
F = 4
F = 5
F = 6
F = 7
F = 8
```

Luego el resultado es el mismo.

6.2. Escribe lo que imprime en la pantalla el siguiente programa, al ser ejecutado:

```
10 LET X = 0
20 FOR K = 1 TO 4
30 LET X = X + K
40 NEXT K
50 PRINT X
60 END
```

La suma de los cuatro primeros números naturales 1, 2, 3 y 4.

```
10
```

6.3. Haz un programa que produzca los mismos resultados en la pantalla que el anterior, sin utilizar las instrucciones **FOR - NEXT.**

```
10  LET X = 0 : LET K = 1
20  LET X = X + K
30  LET K = K + 1
40  IF K < 5 THEN GOTO 20
50  PRINT X
60  END
```

6.4. ¿Qué imprimirá en la pantalla el siguiente programa?

```
10  LET P = 1
20  FOR K = 1 TO 4
30  LET P = P * K
40  NEXT K
50  PRINT P
60  END
```

El producto de los cuatro primeros números naturales 1, 2, 3 y 4.

```
24
```

6.5. Analiza el siguiente programa y escribe lo que imprime en la pantalla:

```
10  LET N = 1
20  FOR K = 1 TO N
30  PRINT "*"
40  PRINT
50  NEXT K
60  LET N = N + 1
70  IF N > 10 THEN GOTO 90
80  GOTO 20
90  END
```

Imprime diez asteriscos separados por líneas en blanco.

```
*
*
*
*
*
*
.
.
.
```

6.6. ¿Qué imprime el siguiente programa?

```
10  LET S = 0
20  FOR K = 1 TO 7 STEP 2
30  LET S = S + K
40  NEXT K
50  PRINT S
60  END
```

Imprime la suma de los números 1, 3, 5 y 7, que es 16.

6.7. ¿Son correctos los programas que incluyen las siguientes instrucciones?

a)
```
20  FOR I = 1 TO 100 STEP 2
..............................................
80  NEXT J
..............................................
```

b)
```
50  FOR N = N 1 TO N 2
..............................................
90  NEXT N
..............................................
120  IF N = 10 THEN GOTO 75
```

a) No es correcto, ya que la instrucción **80 NEXT J** es incorrecta, ya que la variable del bucle es I. La instrucción correcta es **80 NEXT I.**

b) No es correcto debido a que por medio de la instrucción **120** se entra en el bucle **50 – 90** sin pasar por la primera instrucción de dicho bucle.

6.8. Analiza los siguientes programas e indica si son correctos:

a)
```
100  FOR K = 3 TO –3 STEP –1
..............................................
130  PRINT X ↑ K
..............................................
150  NEXT K
160  END
```

b)
```
25  FOR X = 1 TO 2 STEP 0.05
..............................................
50  FOR Y = 1 TO 10 STEP 0.1
..............................................
75  NEXT X
..............................................
100  NEXT Y
..............................................
```

c)
```
100   FOR I = 1 TO M
..............................................
125   FOR J = 1 TO N
..............................................
135   FOR K = 1 TO M + N
..............................................
160   NEXT K
..............................................
180   NEXT J
..............................................
200   FOR J = 1 TO N
..............................................
225   NEXT J
```

d)
```
10   FOR A = 5 TO 10
20   FOR B = 1 TO 3
..............................................
60   NEXT B
70   NEXT A
..............................................
100   FOR B = 1 TO 4
110   FOR A = 4 TO 10
..............................................
140   NEXT B
..............................................
180   NEXT A
..............................................
235   FOR K = 1 TO M + N
..............................................
240   NEXT K
..............................................
250   NEXT I
..............................................
```

a) Es correcto. (En algunos microordenadores es necesario dar algún valor a la variable X, antes de operar con ella.)

b) No es correcto, debido a que los bucles no están uno dentro del otro.

c) Es correcto pues los bucles están bien anidados.

d) Es incorrecto pues la instrucción **140** debería ser **140 NEXT A** y la **180, 180 NEXT B,** para que los bucles estuvieran bien anidados.

6.9. Escribe un programa que calcule e imprima la suma de los números enteros comprendidos entre 1 y N, ambos inclusive, siendo N un valor que se introduce mediante una instrucción **INPUT.**

```
 1   REM SUMA DE LOS N PRIMEROS NUMEROS ENTEROS
10   S = 0
20   INPUT N
30   FOR I = 1 TO N
40   LET S = S + I
50   NEXT I
60   PRINT S
70   END
```

6.10. Escribe un programa que confeccione una tabla de las raíces quintas de los números comprendidos entre 1 y 2, en intervalos de 0.1.

```
 1 REM RAICES QUINTAS
10 R = 0
20 FOR I = 1 TO 2 STEP 0.1
30 LET R = I ↑ (1/5)
40 PRINT R
50 NEXT I
60 END
```

6.11. Haz un programa que calcule la suma

$1^2 + 2^2 + 3^2 + ... + 99^2$

```
10 LET S = 0
20 FOR I = 1 TO 99
30 LET S = S + I ↑ 2
40 NEXT I
50 PRINT S
60 END
```

6.12. Escribe un programa que calcule la media aritmética de una lista de N números utilizando una instrucción **INPUT** para introducir N y después un bucle **FOR - NEXT** en el que esté incluida otra instrucción **INPUT** o una **READ** para introducir cada número.

```
10 LET S = 0
20 INPUT N
30 FOR I = 1 TO N
40 INPUT A
50 LET S = S + A
60 NEXT I
70 PRINT "LA MEDIA ES:"; S/N
80 END
```

En lugar de la instrucción **40** podría utilizarse **40 READ A,** añadiendo las instrucciones **DATA** necesarias.

6.13. Elabora un programa que calcule el producto de los N primeros números naturales, sin contar el cero. (Observación: El producto aludido en este enunciado se llama factorial de N y suele simbolizarse así: N!. Por ejemplo, el producto 1 * 2 * 3 * 4 * 5 * 6 * 7 se llama factorial de 7 y se indica por 7!.)

```
 1 REM PRODUCTO DE LOS N PRIMEROS NUMEROS
10 INPUT N
20 LET P = 1
30 FOR I = 1 TO N
40 LET P = P * I
50 NEXT I
60 PRINT "EL FACTORIAL DE"; N; "ES"; P
70 END
```

7 Cadenas

Programas resueltos con LEN, VAL, STR$, LEFT$, RIGHT$ y MID$

RESUMEN DE LAS INSTRUCCIONES

CADENAS DE CARACTERES

Una cadena es una sucesión de letras, números y cualquier otro símbolo gráfico.

Las cadenas se escriben entre comillas, para indicar el principio y el fin de las mismas.



Ejemplos

- "A 3 2 2 ?"
- "LOPEZ"
- "   "

  Este último ejemplo también es una cadena, pues el espacio en blanco también se considera como carácter.
- "CARMEN"
- "EN CIERTO LUGAR DE LA MANCHA"
- "323"

  Una secuencia de dígitos dispuestos entre comillas constituye una cadena, y no un número.

Observación

Así como los números se guardan en variables numéricas, las cadenas se guardan en variables de cadena o alfanuméricas. Se distinguen de las numéricas en que su nombre termina con el símbolo $. Ejemplos: A$, X$, Z3$.

SÍMBOLOS >, <, < >, =

Con la ayuda de estos símbolos puede compararse cadenas.



Ejemplos

- "NABUCODONOSOR" < > "LOPEZ"

  La cadena primera **es distinta a** la segunda.
- A$ = "LOPEZ"
  La variable de cadena A$ **es igual a** LOPEZ.
- "LOPEZ" < "PEREZ"
  "LOPEZ" **es anterior a** "PEREZ" en el orden alfabético, pues la letra L es anterior a la P.
- "SASTRE" > "SANCHEZ"
  La comparación de cadenas se hace carácter a carácter, de izquierda a derecha. Luego, como el tercer carácter de la primera cadena (S) es posterior al tercero de la segunda cadena, la primera cadena **es posterior a** la segunda.
- "A1" < "A2"
  Si las cadenas contienen cifras, en la comparación se tiene en cuenta el orden numérico. De ahí que "A1" sea **anterior a** "A2".

OPERACIÓN +

Dos o más cadenas se pueden **sumar** o **concatenar,** y así formar una más larga. Esta operación se indica con el signo +.



Ejemplos

- Después de ejecutarse esta instrucción

  ```
  10 LET C$ = "CARMEN" + "PEREZ"
  ```

  la variable C$ almacena la cadena "CARMEN PEREZ".
- Al terminar de ejecutarse este programa

  ```
  10 LET A$ = "LA VIDA ES"
  20 LET B$ = " "
  30 LET C$ = "HERMOSA"
  40 LET D$ = A$ + B$ + C$
  ```

  la variable D$ guarda la cadena "LA VIDA ES HERMOSA".

LEN

Esta función da el número de caracteres contenido en una determinada cadena.



Ejemplo

Al ejecutar estas dos instrucciones

```
10  LET A$ = "NABUCODONOSOR"
20  PRINT LEN (A$)
```

se imprime en la pantalla el número 12, número de caracteres de la palabra contenida en A$.

VAL

Esta función convierte una cadena numérica en un número.



Ejemplo

Al ejecutarse este programa

```
10  LET A$ = "30"
20  LET X = VAL (A$)
30  PRINT X
```

se imprime en pantalla el contenido de la variable numérica X, que es 30.

Inicialmente, "30" era una cadena que se almacenaba en la variable de cadena A$.

STR$

La función STR$ convierte un número en cadena de caracteres.



Ejemplo

Después de ejecutarse este programa

```
10  LET X = 12310
20  LET X$ = STR$ (X)
30  PRINT X$
```

se imprime en la pantalla el contenido de la variable de cadena IX formada por un conjunto de caracteres que inicialmente constituían un número que se almacenaba en la variable numérica.

LEFT$

Esta función extrae los primeros caracteres de una cadena.



Ejemplo

```
10  LET A$ = "CIRCUNSTANCIA"
20  PRINT LEFT$ (A$, 4)
```

La instrucción **10** asigna a A$ la cadena "CIRCUNSTANCIA" y la **20** imprime en la pantalla los cuatro primeros caracteres de la cadena A$; es decir, imprime CIRC.

RIGHT$

La función RIGHT extrae los últimos caracteres de la cadena A$.



Ejemplo

```
10  LET A$ = "CIRCUNSTANCIA"
20  PRINT RIGHT$ (A$, 3)
```

La instrucción **20** imprime en la pantalla los tres últimos caracteres de la cadena A$; es decir, imprime CIA.

MID$

Extrae un cierto número de caracteres de una cadena a partir de una posición.



Ejemplo

```
10  LET A$ = "CIRCUNSTANCIA"
20  PRINT MID$ (A$, 2, 3)
```

La instrucción **20** imprime en la pantalla tres caracteres a partir del que ocupa la posición 2; es decir, imprime IRC.

TO

Mediante la palabra TO, algunos microordenadores extraen los caracteres comprendidos entre dos determinados.



Ejemplos

- **10 LET A$ = "CIRCUNSTANCIA"**
  20 PRINT A$ (2 TO 5)

  La instrucción **20** imprime en la pantalla desde el carácter 2 al carácter 5 de la cadena A$; es decir, imprime IRCUN.

- **10 LET A$ = "CIRCUNSTANCIA"**
  20 PRINT A$ (2 TO)

  La instrucción **20** imprime desde el carácter 2 hasta el final de la cadena; es decir, imprime CIRCUNSTANCIA.

- **10 LET A$ = "CIRCUNSTANCIA"**
  20 PRINT A$ (TO 5)

  La instrucción **20** imprime los caracteres de la cadena A$ comprendidos entre el primero y el quinto, ambos inclusive; es decir, imprime CIRCU.

- **10 LET A$ = "CIRCUNSTANCIA"**
  20 PRINT A$ (6 TO 6)

  La instrucción **20** imprime en la pantalla el carácter sexto de la cadena A$; es decir, imprime la N.

RESUMEN DE OTRAS INSTRUCCIONES

CLS

Esta instrucción borra la pantalla del microordenador.



Ejemplo

Al ejecutarse este programa

```
10 PRINT "UN ROBOT TAMBIEN TIENE CORAZONCITO"
20 CLS
```

en la pantalla aparece muy brevemente la frase escrita en la instrucción **10**, ya que enseguida la borra la instrucción **20**.

PAUSE O WAIT

Estas instrucciones detienen la ejecución del programa y retienen la imagen de la pantalla un cierto tiempo.



Ejemplo

```
10 PRINT "UN ROBOT TIENE SU CORAZONCITO"
20 PAUSE 20
30 CLS
```

La instrucción **20** detiene la imagen un breve tiempo y la instrucción **30** borra dicha imagen.

Observación

Algunos microordenadores disponen de la instrucción **PAUSE**, otros disponen de **WAIT** y otros carecen de esta instrucción.

En estos últimos, para conseguir los mismos efectos, se puede utilizar un bucle como el siguiente (bucle de retardo):

```
10 FOR I = 1 TO 1000
20 NEXT I
```

El tiempo de espera depende del valor tope del bucle; en este ejemplo, el tope es 1000, que produce un tiempo de espera de aproximadamente dos segundos.

PROGRAMAS RESUELTOS EXPLICADOS

7.1. Si al ejecutar este programa introduces tu nombre, ¿qué obtendrás en la pantalla?

```
10 INPUT "NOMBRE"; A$
20 FOR I = 1 TO 10
30 PRINT "EL ALUMNO"; A$; "ES UN GRAN PROGRAMADOR"
40 NEXT I
50 END
```

Si se introduce, por ejemplo, el nombre LUIS, escribirá 10 veces este comentario:

EL ALUMNOLUISES UN GRAN PROGRAMADOR

Para evitar que el nombre del alumno salga unido a las palabras ALUMNO y ES puede modificarse la instrucción **30**, escribiéndola así:

```
30 PRINT "EL ALUMNO"; A$; "ES UN GRAN PROGRAMADOR"
```

7.2. Haz un programa que lea el nombre y los apellidos de una persona y los imprima infinitas veces.

```
10 INPUT "NOMBRE"; N$
20 INPUT "APELLIDOS"; A$
30 PRINT N$; " "; A$
40 GOTO 30
50 END
```

Una vez introducidos el nombre y los apellidos, la instrucción **30** los imprime separados por un espacio en blanco.

La instrucción **40** envía el control a la **30**, lo que hace que el proceso se repita indefinidamente.

7.3. Escribe un programa que lea los días de la semana en una instrucción **DATA** y los imprima seguidos, dejando un espacio en blanco entre cada dos nombres.

```
 10 FOR I = 1 TO 7
 20 READ D$
 30 PRINT D$; " ";
 40 NEXT I
 50 END
100 DATA LUNES, MARTES, MIERCOLES, JUEVES, VIERNES,
    SABADO, DOMINGO
```

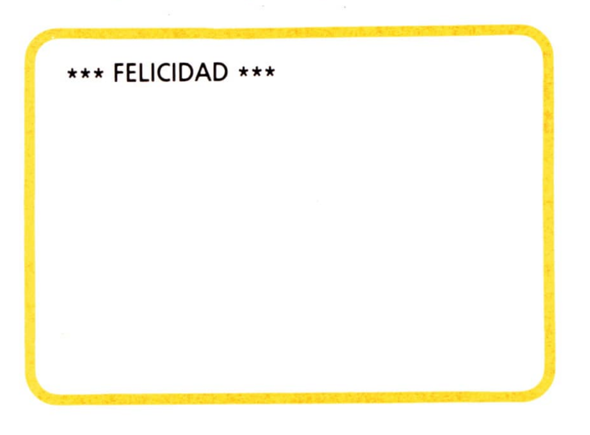
El proceso de lectura se repite 7 veces, imprimiéndose en cada una de ellas un día de la semana y un espacio en blanco. El punto y coma al final de la instrucción **30** hace que el nombre de cada día se imprima a continuación de un espacio en blanco.

7.4. ¿Qué imprime en la pantalla el siguiente programa si cuando se ejecuta se introduce una palabra?

```
10 INPUT A$
20 LET X$ = "***" + A$ + "***"
30 PRINT X$
40 END
```

Si al ejecutarse la instrucción **10 INPUT A$** se introduce, por ejemplo, la cadena FELICIDAD, en la pantalla aparece la palabra

FELICIDAD precedida de tres asteriscos y seguida también de tres asteriscos, ya que la cadena A$ se concatena con tres asteriscos por delante y tres por detrás.



7.5. Haz un programa que permita introducir dos palabras, las concatene intercalando cinco asteriscos entre ellas e imprima el resultado.

```
10  INPUT A$
20  INPUT B$
30  LET X$ = A$ + "*****" + B$
40  PRINT X$
50  END
```

La variable X$ se forma concatenando la cadena A$, 5 asteriscos y la cadena B$. La instrucción **40** imprime esta cadena X$.

7.6. Diseña un programa que dé las longitudes de las cadenas que se van introduciendo y que detenga su ejecución al introducir una de longitud 1.

```
10  INPUT A$
20  LET L = LEN (A$)
30  PRINT "LA LONGITUD DE "; A$; "ES"; L
40  IF L < > 1 THEN GOTO 10
50  END
```

— El microordenador obtiene la longitud de la cadena en la instrucción **20** e imprime dicha longitud en la instrucción **30.**
— En la instrucción **40** se analiza si esta longitud es distinta de uno, en cuyo caso se lee una nueva cadena y en caso contrario se detiene la ejecución del programa.

7.7. Realiza un programa que permita introducir como cadena el nombre de un abonado y como número, su número de teléfono; transforme éste en cadena, concatene ambas cadenas y, finalmente, imprima el resultado.

```
10 INPUT "INTRODUCE EL NOMBRE"; N$
20 INPUT "INTRODUCE EL TELEFONO"; T
30 LET T$ = STR$ (T)
40 LET X$ = N$ + " " + T$
50 PRINT X$
60 END
```

Después de introducir el nombre para N$, se introduce el número de teléfono, que se asigna a la variable numérica T.

La instrucción **30 LET T$ = STR$ (T)** transforma el número en cadena, y lo asigna a la variable T$.

En la instrucción **40** se concatenan N$, un espacio en blanco y la cadena T$. La instrucción **50** imprime el contenido de X$.

7.8. Haz un programa que forme una subcadena con la primera y última letra de cualquier cadena que se introduzca a través del teclado.

```
10 INPUT A$
20 LET I$ = LEFT$ (A$, 1)
30 LET D$ = RIGHT$ (A$, 1)
40 LET S$ = I$ + D$
50 PRINT S$
60 END
```

Las instrucciones **20** y **30** extraen el primer y último carácter de la cadena A$ y los almacenan en I$ y D$.

Estas dos cadenas se concatenan en la instrucción **40,** formando S$, y su contenido se imprime en la pantalla mediante la instrucción **60.**

En el microordenador ZX Spectrum, un programa que produciría los mismos resultados podría ser éste:

```
10 INPUT A$
20 LET L = LEN (A$)
20 LET I$ = A$ (1 TO 1)
30 LET D$ = A$ (L TO L)
40 LET S$ = I$ + D$
50 PRINT S$
60 END
```

7.9. Elabora un programa que lea una cadena, calcule su longitud L e imprima en la primera fila el primer carácter L veces; en la segunda fila, el segundo carácter L veces, y así sucesivamente.

```
 10  INPUT A$
 20  LET L = LEN (A$)
 30  FOR I = 1 TO L
 40  LET X$ = MID$ (A$, I, 1)
 50  FOR J = 1 TO L
 60  PRINT X$;
 70  NEXT J
 80  PRINT
 90  NEXT I
100  END
```

— Averiguada la longitud de la cadena A$ en la instrucción **20,** se obtiene cada uno de sus caracteres en la instrucción **40,** según va cambiando de valor la variable I.

— Cada uno de estos caracteres se imprime L veces en la pantalla, mediante el bucle que va de la instrucción **30** a la **70.**

— La instrucción **80 PRINT** sirve para dejar una línea entre las distintas impresiones de cada carácter.

En el microordenador ZX Spectrum debe sustituirse la instrucción **40** por

```
40  LET X$ = A$ (I TO I)
```

7.10. Realiza un programa que lea cadenas y las imprima junto con su longitud hasta que se introduzca la cadena "BASTA".

```
10  INPUT A$
20  IF A$ = "BASTA" THEN GOTO 60
30  LET L = LEN (A$)
40  PRINT A$, L
50  GOTO 10
60  END
```

La instrucción **20** permite analizar si se ha introducido la cana "BASTA", en cuyo caso se interrumpe la ejecución del programa. En caso contrario, se calcula la longitud de la cadena, y se imprimen ésta y dicha longitud.

A continuación se pide una nueva cadena.

7.11. Construye un programa que imprima el primer carácter de la cadena que se introduzca si ésta es anterior a "ULTIMA", y termine su ejecución cuando se introduzca "FIN".

```
10 LET I$ = " "
20 INPUT A$
30 IF A$ = "FIN" THEN GOTO 70
40 IF A$ = "ULTIMA" THEN PRINT I$
50 LET I$ = LEFT$ (A$, 1)
60 GOTO 20
70 END
```

En la cadena I$ se almacena el primer carácter de la cadena A$ (inicialmente I$ contiene la cadena vacía).

Antes de asignar el primer carácter a I$ se comprueba si A$ es igual a "ULTIMA", en cuyo caso se imprime el contenido anterior de I$. Si A$ es "FIN", el proceso termina.

En el microordenador ZX Spectrum debe sustituirse la instrucción **50** por

```
50 LET I$ = A$ (1 TO 1)
```

PROGRAMAS RESUELTOS

7.1. ¿Qué caracteres imprime el siguiente programa?

```
10 LET A$ = "FERNANDEZ"
20 PRINT LEFT$ (A$, 2)
30 PRINT RIGHT$ (A$, 1)
40 PRINT MID$ (A$, 2, 3)
50 END
```

```
FE

Z

ERN
```

7.2. Escribe las instrucciones necesarias para imprimir el número de caracteres de la palabra NABUCODONOSOR.

```
10  LET N = LEN ("NABUCODONOSOR")
20  PRINT N
30  END
```

7.3. ¿Qué imprimirá el siguiente programa?

```
10  LET A$ = "PATA"
20  LET B$ = "TA"
30  PRINT A$ + B$
40  END
```



7.4. ¿Qué instrucciones son incorrectas? ¿Por qué?

a) 100 IF X = "FECHA" THEN GOTO 500
b) 200 IF N$ < > A + B THEN PRINT N$
c) 300 IF P$ = "1234" THEN LET S = S + 1
d) 400 IF G$ = "MAYOR" THEN GOTO 600
e) 500 IF A$ = "B" AND X > 3 THEN GOTO 800

a) No es correcta, pues X no es variable de cadena.
b) No es correcta, pues se comparan variables numéricas y de cadena.
c) Es correcta, pues al estar 1234 entre comillas se considera una cadena.
d) Es correcta.
e) Es correcta.

7.5. ¿Qué tiene que ocurrir para que se cumpla la condición P$ > Q$?

Que la cadena que contiene la variable P$ sea anterior a la que contiene Q$.

7.6. Indica si son correctas las siguientes relaciones entre cadenas:

a) **"PEDRO" > = "JUAN"**
b) **"M33" < "M4"**
c) **"127" > "1235"**
d) **"4PERRO" < = "37GATO"**

a) Es correcta, pues P es posterior a J.
b) Es correcta, pues aunque M esté en ambas cadenas, 3 es anterior a 4.
c) Es correcta, pues aunque los dos primeros caracteres coinciden, el 7 es posterior al 3.
d) Es incorrecta, pues 4 es posterior a 3.

7.7. Escribe una o varias instrucciones que expresen la siguiente condición:

Si N$ = "SI" transferir el control a la línea **70,** y si N$ = "NO" transferirlo a la **90.**

```
10  IF N$ = "SI" THEN GOTO 70
20  IF N$ = "NO" THEN GOTO 90
```

7.8. Sea una cadena A$ en la que están almacenados el primer apellido de un alumno, a continuación el segundo apellido y después el nombre, cada uno con diez caracteres (si algunos no se ocupan se dejan con blancos); además está almacenada la edad, que ocupa dos caracteres. Escribe un programa que imprima por separado cada una de estas partes de la cadena.

```
 10  INPUT A$
 20  LET B$ = LEFT$ (A$, 10)
 30  LET C$ = MID$ (A$, 11, 10)
 40  LET D$ = MID$ (A$, 21, 10)
 50  LET E$ = RIGHT$ (A$, 2)
 60  PRINT "1 APELLIDO", B$
 70  PRINT "2 APELLIDO", C$
 80  PRINT "NOMBRE", D$
 90  PRINT "EDAD", E$
100  END
```

Para el ZX Spectrum sustituir las instrucciones **20, 30, 40** y **50** por

```
20  LET B$ = A$ (1 TO 10)
30  LET C$ = A$ (11 TO 20)
40  LET D$ = A$ (21 TO 30)
50  LET E$ = A$ (31 TO 32)
```

7.9. Escribe un programa tal que para tres personas cuyos apellidos, nombres y edad se conocen y están incluidos en tres cadenas A$, B$ y C$, imprima

en pantalla su edad media (ver el ejercicio anterior y usar la instrucción VAL (X$)).

```
 10  INPUT A$
 20  INPUT B$
 30  INPUT C$
 40  LET A$ = RIGHT$ (A$, 2)
 50  LET B$ = RIGHT$ (B$, 2)
 60  LET C$ = RIGHT$ (C$, 2)
 70  LET X = VAL (A$)
 80  LET Y = VAL (B$)
 90  LET Z = VAL (C$)
100  PRINT (X + Y + Z) / 3
110  END
```

Para el ZX Spectrum añadir la instrucción

```
35  LET L1 = LEN (A$) : LET L2 = LEN (B$) : LET L3 = LEN (C$)
```

y sustituir las instrucciones **40**, **50** y **60** por

```
40  LET A$ = A$ (L1-1 TO L1)
50  LET B$ = A$ (L2-1 TO L2)
60  LET C$ = A$ (L3-1 TO L3)
```

7.10. Haz un programa que introduzca por separado el día, el mes y el año de nacimiento de una persona, forme una única cadena para estos datos e imprima el resultado.

```
10  INPUT D$
20  INPUT M$
30  INPUT A$
40  LET C$ = D$ + M$ + A$
50  PRINT C$
60  END
```

7.11. Escribe un programa que permita introducir una palabra de cinco letras y después imprima sus caracteres de derecha a izquierda.

```
10  INPUT A$
20  LET B$ = " "
30  FOR I = 5 TO 1 STEP -1
40  LET B$ = B$ + MID$ (A$, I, 1)
50  NEXT I
60  PRINT B$
70  END
```

Para el ZX Spectrum, cambiar la instrucción **40** por

```
40  LET B$ = B$ + A$ (I TO I)
```

7.12. Elabora un programa que invierta una palabra con cualquier número de letras.

```
10  INPUT A$
20  LET B$ = "  "
25  LET L = LEN (A$)
30  FOR I = L TO 1 STEP -1
40  LET B$ = B$ + MID$ (A$, I, 1)
50  NEXT I
60  PRINT B$
70  END
```

Para el ZX Spectrum cambiar la instrucción **40** por

```
40  LET B$ = B$ + A$ (I TO I)
```

7.13. Realiza un programa que lea una serie de palabras e imprima solamente las que empiecen por la letra A.

```
10  INPUT A$
20  IF A $ < > LEFT $ (A$, 1) THEN 10
30  PRINT A$
40  GOTO 10
50  END
```

7.14. Diseña un programa que lea una serie de palabras e imprima solamente las que empiecen por la letra A y terminen con la S.

```
10  INPUT A$
20  IF (A $ < > LEFT $ (A$, 1)) OR (S < > RIGHT$ (A$, 1)) THEN
     GOTO 10
30  PRINT A$
40  GOTO 10
50  END
```

7.15. Dadas las variables:

A$ = "EL"
B$ = "DIA"
C$ = "COMIENZA"
D$ = "AL"
E$ = "AMANECER"

hacer un programa que realice la concatenación de las cadenas, que almacene las variables e imprima el resultado.

```
10  INPUT A$
20  INPUT B$
30  INPUT C$
40  INPUT D$
50  INPUT E$
60  LET Z$ = A$ + B$ + C$ + D$ + E$
70  PRINT Z$
80  END
```

7.16. Rectifica el programa anterior para que al imprimir deje un espacio en blanco entre cada dos cadenas.

```
10  INPUT A$
20  INPUT B$
30  INPUT C$
40  INPUT D$
50  INPUT E$
60  LET Z$ = A$ + " " + B$ + " " + C$ + " " + D$ + " " + E$
70  PRINT Z$
80  END
```

7.17. Realiza un programa que lea tres cadenas y dé la longitud total de las tres.

```
10  INPUT A$
20  INPUT B$
30  INPUT C$
40  LET Z$ = A$ + B$ + C$
50  LET L = LEN (Z$)
60  PRINT L
```

7.18. Construye un programa que lea una cadena, calcule su longitud y vaya extrayendo subcadenas desde la izquierda, de longitud 1, de longitud 2, ..., y de longitud máxima.

```
10  INPUT A$
20  LET L = LEN (A$)
30  PRINT "LA LONGITUD ES", L
40  FOR I = 1 TO L
50  LET B$ = LEFT$ (A$, I)
60  PRINT B$
70  NEXT I
80  END
```

Para el ZX Spectrum cambiar la instrucción **50** por

```
50  LET B$ = A$ (1 TO I)
```

7.19. Haz un programa que lea una cadena y añade blancos (espacios en blanco) hasta que alcance la longitud 20.

```
10  INPUT A$
20  LET L = LEN (A$)
30  FOR I = L + 1 TO 20
40  LET A$ = A$ + " "
50  NEXT I
60  END
```

7.20. Realiza un programa que dé el informe "CORRECTO" si la cadena leída tiene longitud inferior a 3 y que imprima "MAL" en caso contrario.

```
10  INPUT A$
20  LET L = LEN (A$)
30  IF L < 3 THEN GOTO 60
40  PRINT "MAL"
50  STOP
60  PRINT "CORRECTO"
70  END
```

7.21. Diseña un programa que lea cadenas e imprime el primer carácter y el último hasta que se introduzca la cadena "FIN".

```
10  INPUT A$
20  IF A$ = "FIN" THEN GOTO 70
30  LET I$ = LEFT$ (A$, 1) : LET D$ = RIGHT$ (A$, 1)
40  PRINT "EL PRIMER CARACTER ES:"; I$
50  PRINT "EL ULTIMO CARACTER ES:"; D$
60  GOTO 10
70  END
```

En el ZX Spectrum añadir

```
25  LET L = LEN (A$)
```

y sustituir la instrucción **30** por

```
30  LET I$ = A$ (1 TO 1) : LET D$ = A$ (L TO L)
```

7.22. Escribe un programa que compruebe si una cadena contiene la letra A.

```
10  INPUT A$
20  LET L =LEN (A$)
30  FOR I = 1 TO L
40  LET Z$ = MID$ (A$, I, 1)
```

```
50 IF Z$ = "A" THEN GOTO 90
60 NEXT I
70 PRINT "LA CADENA NO CONTINE LA LETRA A"
80 STOP
90 PRINT "LA CADENA CONTIENE LA LETRA A"
100 END
```

En el ZX Spectrum, sustituir la instrucción **40** por

```
40 LET Z$ = A$ (I TO I)
```

7.23. Diseña un programa tal que al introducir un texto calcule el número de veces que aparece la letra E.

```
10 LET C = 0
20 INPUT A$
30 LET L = LEN (A$)
40 FOR I = 1 TO L
50 LET B$ = MID$ (A$, I, 1)
60 IF B$ = "E" THEN LET C = C + 1
70 NEXT I
80 PRINT "LA LETRA E ESTA CONTENIDA "; C; " VECES"
90 END
```

(En muchos microordenadores el texto que se introduce en A$ sólo puede tener como máximo 255 caracteres.)

En el ZX Spectrum sustituir la instrucción **50** por

```
50 LET B$ = A$ (I TO I)
```

7.24. Escribe un programa que cuente el número total de vocales que hay en un texto cualquiera.

```
10 LET C = 0
20 INPUT A$
30 LET L = LEN (A$)
40 FOR I = 1 TO L
50 LET B$ = MID$ (A$, I, 1)
60 IF B$ = "A" OR B$ = "E" OR B$ = "I" OR B$ = "O" OR B$ =
   = "U" THEN LET C = C + 1
70 NEXT I
80 PRINT "HAY "; C; " VOCALES"
90 END
```

En el Spectrum hay que sustituir la instrucción **50** por

```
50 LET B$ = A$ (I TO I)
```

7.25. Haz un programa que lea un texto y dé un informe con el número de veces que han aparecido cada una de las vocales: A, E, I, O y U.

```
10  LET A = 0 : LET E = 0 : LET I = 0 : LET 0 = 0 : LET U = 0
20  INPUT A$
30  LET L = LEN (A$)
40  FOR K = 1 TO L
50  LET B$ = MID$ (A$,K,1)
60  IF B$ = "A" THEN LET A = A + 1
70  IF B$ = "E" THEN LET E = E + 1
80  IF B$ = "I" THEN LET I = I + 1
90  IF B$ = "O" THEN LET 0 = 0 + 1
100 IF B$ = "U" THEN LET U = U + 1
110 NEXT K
120 PRINT "LA LETRA A APARECE "; A; " VECES"
130 PRINT "LA LETRA E APARECE "; E; " VECES"
140 PRINT "LA LETRA I APARECE "; I; " VECES"
150 PRINT "LA LETRA O APARECE "; O; " VECES"
160 PRINT "LA LETRA U APARECE "; U; " VECES"
170 END
```

8 Listas numéricas

Programas resueltos con DIM A(N)

RESUMEN DE LAS INSTRUCCIONES

DIM A(N)

Esta instrucción reserva posiciones de memoria, en las que se almacenarán los datos de las variables.

Inicialmente, en estas posiciones se almacena el valor cero.



Ejemplo

La instrucción

```
10  DIM A(20)
```

reserva, según el microordenador,

— 20 posiciones de memoria si empieza a contar desde el 1. Estas posiciones se nombran así con los nombres A(1), A(2), ..., A(20).

— 21 posiciones de memoria, si empieza a contar desde el 0. Estas posiciones se nombran con los nombres A(0), A(1), ..., A (20).

Observación

En algunos microordenadores, cuando N es menor o igual que 10, no es necesario utilizar la instrucción **DIM**, pues la reserva de posiciones de memoria se realiza de forma automática.

ERASE

Esta instrucción elimina la declaración de dimensión de las listas especificadas, dejando libres las posiciones de memoria reservadas y los nombres de dichas listas.



Ejemplo

Al ejecutarse la instrucción

```
40  ERASE A, B
```

las variables A y B quedan libres de su anterior dimensionamiento.

Observación

¡No todos los microordenadores poseen esta instrucción!

PROGRAMAS RESUELTOS EXPLICADOS

8.1. Indica cuántas posiciones de memoria abarca la variable dimensionada A(K).

```
10  DIM A(18)
20  FOR K = 1 TO 15
30  INPUT A(K)
40  NEXT K
50  END
```

La variable A(K) ocupa 18 ó 19 posiciones (si el microordenador empieza a contar desde 0) porque la instrucción **10 DIM A(18)** así lo especifica, aunque el programa sólo utilice quince posiciones (de A(1) a A(15)).

8.2. ¿Qué ocurriría al ejecutar el programa anterior si la instrucción **10 DIM A(18)** se cambiara por **10 DIM A(13)?**

El microordenador imprimiría un mensaje de error en el momento de su ejecución, al ser el número de posiciones reservadas inferior al número que se pretende utilizar.

8.3. Haz un programa que produzca los mismos efectos que el transcrito en el ejercicio 8.1, utilizando las instrucciones **READ** y **DATA**.

```
10 DIM A(18)
20 FOR K = 1 TO 15
30 READ A(K)
40 NEXT K
50 DATA ...
60 END
```

Solamente ha sido necesario sustituir la instrucción **30 INPUT A(K)** por **30 READ A(K)** y añadir la instrucción de datos **50 DATA ...** donde deberán ir quince valores separados por comas.

8.4. Escribe un programa que lea una lista de 12 datos numéricos; es decir, los almacene en la memoria y los imprima en la pantalla uno a continuación de otro, en fila.

```
10 DIM A(12)
20 FOR I = 1 TO 12
30 INPUT A(I)
40 NEXT I
50 FOR I = 1 TO 12
60 PRINT A(I);
70 NEXT I
80 END
```

— La instrucción **10** reserva 12 posiciones de memoria para la variable A de un índice. (13 posiciones en los microordenadores que cuenten desde 0).
— El bucle **20 - 40** permite almacenar en la memoria los 12 datos numéricos.
— El bucle **50 - 70** imprime los valores en la pantalla uno a continuación de otro debido al punto y coma (;) de la instrucción **60.**

8.5. Escribe una lista de diez números y ordénalos siguiendo las instrucciones del programa anterior.

```
 5 REM ORDENACION DE NUMEROS
10 INPUT "NUMERO:"; N
20 DIM V(N)
30 FOR J = 1 TO N
40 INPUT "DATO:"; V(J)
50 NEXT J
```

```
60  FOR J = 1 TO N - 1
70  FOR K = J + 1 TO N
80  IF V(J) < = V(K) THEN GOTO 120
90  LET A = V(J)
100 LET V(J) = V(K)
110 LET V(K) = A
120 NEXT K
130 NEXT J
140 FOR I = 1 TO N
150 PRINT V(I); " ";
160 NEXT I
170 END
```

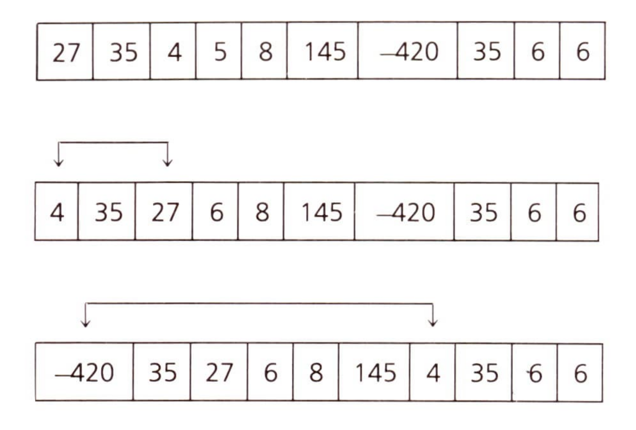
Si se introducen los números 27, 35, 4, 6, 8, 145, –420, 35, 6, 6, el resultado que se obtiene es éste:

```
–420  4  6  6  6  8  27  35  35  145
```

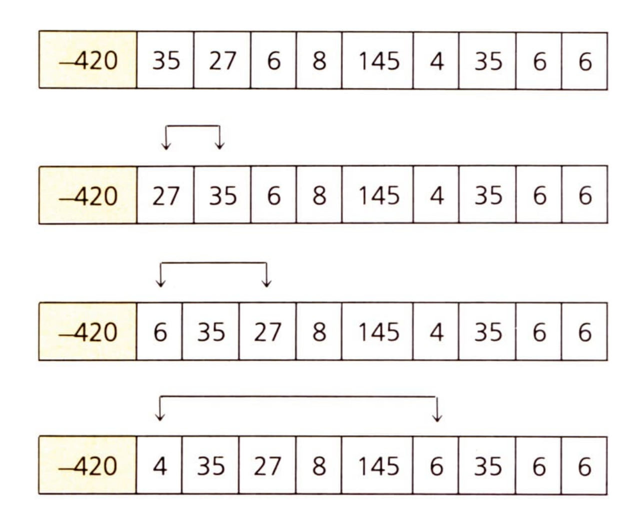
El conjunto de instrucciones que hacen la ordenación está comprendido entre la **60** y la **130,** ambas inclusive, y el método utilizado se explica a continuación.

— **Primera fase (J = 1): llevar a la primera posición el número menor.**

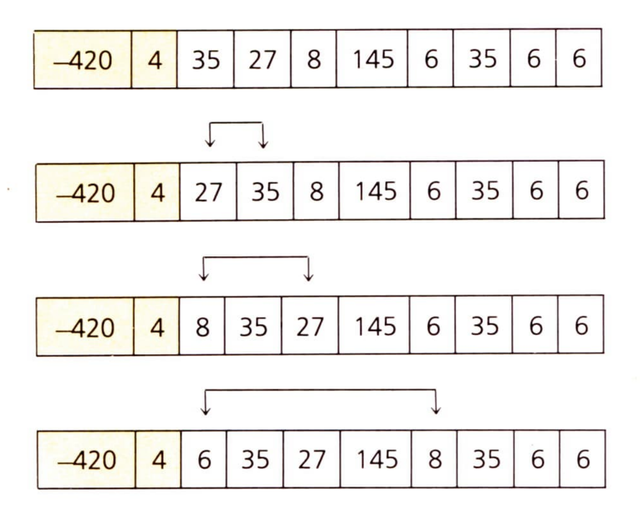
Para conseguir esto se compara la primera posición con todas las siguientes; si éstas contienen números menores que el de la primera posición, se intercambian los valores:



— **Segunda fase (J = 2): llevar a la segunda posición el número menor (mayor que el primero).**



— **Tercera fase (J = 3): llevar a la tercera posición el número menor (mayor que el segundo).**



— Y así sucesivamente, hasta conseguir que todos los números de la lista queden ordenados.

Se obtiene este resultado:

–420	4	6	6	6	8	27	35	35	145

8.6. Haz un programa que ordene una lista de números de mayor a menor, o sea en forma decreciente, y los escriba en la pantalla en forma vertical.

```
  1 REM ORDENACION DE NUMEROS DE MAYOR A MENOR
 10 INPUT "NUMERO:"; N
 20 DIM V(N)
 30 FOR J = 1 TO N
 40 INPUT "DATO:"; V(J)
 50 NEXT J
 60 FOR J = 1 TO N - 1
 70 FOR K = J + 1 TO N
 80 IF V(J) > = V(K) THEN GOTO 120
 90 LET A = V(J)
100 LET V(J) = V(K)
110 LET V(K) = A
120 NEXT K
130 NEXT J
140 FOR I = 1 TO N
150 PRINT V(I)
160 NEXT I
170 END
```

— El conjunto de instrucciones **10 - 50** permite introducir el número de valores N, y los datos V(J).
— El bloque **60 - 130** realiza la ordenación de los números de mayor a menor, efectuando N - 1 fases (desde J = 1 hasta J = = N - 1), de tal forma que en la primera fase (J = 1) se lleva el número mayor a la primera posición; en la segunda fase (J = = 2) se lleva el número mayor (menor que el primero) y así sucesivamente.
— El bloque **140 - 160** imprime, en forma vertical, los resultados.

PROGRAMAS RESUELTOS

8.1. En la memoria de un microordenador se tienen estas variables con sus correspondientes valores almacenados:

Q	2	A(1)	37
A	3	A(2)	4
A1	1	A(3)	23
		A(4)	19

Escribe el valor que almacenan las siguientes variables: A(A), A(A1), A(A(Q)), A(A(2)).

A(A) = A(3) = 23
A(A1) = A(1) = 37
A(A(Q)) = A(A(2)) = A(4) = 19
A(A(2)) = A(4) = 19

8.2. ¿Qué aparecerá en pantalla al ejecutar el siguiente programa?

```
10  READ N
20  FOR K = 1 TO N
30  READ X(K)
40  NEXT K
50  FOR K = 1 TO N
60  IF X(K) < 0 THEN GOTO 80
70  PRINT X(K)
80  NEXT K
90  DATA 7
100 DATA 23, -44, 37, 0, -14, -38, 14
200 END
```

— Como el valor que se le asigna a N es 7, en algunos microordenadores aparecerá un mensaje de error por no haber dimensionado la lista X(K) al principio del programa con la instrucción

```
15  DIM X(N)
```

— En los que no se produzca error, imprimirán los números no negativos en columna, es decir:

```
23
37
 0
14
```

8.3. Si se añadiesen al programa anterior las siguientes instrucciones:

```
110 DATA 8, 4, -3, 2, 2, 4, 3, 8, 5, 1
120 DATA 5, 0, -2, 4, 3, 9, -8, 4, 3, 8, 4
```

¿qué problemas se presentarían? ¿Cómo se solucionarían?

— Al añadir 21 datos se produciría un error por no haber dimensionado la variable X(K) y por ser N = 7.

— Este problema se soluciona añadiendo al programa la instrucción

```
15  DIM X (N)
```

y sustituyendo el valor 7 (que se asignó a N) por 28:

```
90  DATA 28
```

8.4. Haz un programa que calcule la suma de los 100 primeros elementos de una lista T (I).

```
10  DIM T(100)
20  LET S = 0
30  FOR I = 1 TO 100
40  INPUT T(I)
50  LET S = S + T(I)
60  NEXT I
70  PRINT S
80  END
```

8.5. Dado un conjunto de valores, elabora un programa que escriba aquellos que se encuentren entre dos dados.

El programa debe comenzar leyendo el menor, el mayor y el número de términos. Acto seguido leerá uno a uno los distintos valores e imprimirá los que cumplan la condición.

```
 10  INPUT "EL MENOR"; N
 20  INPUT "EL MAYOR"; M
 30  INPUT "NUMERO DE VALORES"; V
 40  DIM X(V)
 50  FOR I = 1 TO V
 60  INPUT X(I)
 70  NEXT I
 80  FOR I = 1 TO V
 90  IF N < X(I) AND X(I) < M THEN PRINT X(I)
100  NEXT I
110  END
```

8.6. Se tiene una lista de 20 personas entre los 30 y 69 años. Realiza un programa que contabilice el número de personas que se encuentran entre:

30-39
40-49
50-59
60-69

e imprime los resultados.

```
10  DIM P(20)
20  LET A = 0 : LET B = 0 : LET C = 0 : LET D = 0
30  FOR I = 1 TO 20
40  INPUT P(I)
50  NEXT I
60  FOR I = 1 TO 20
70  IF 30 < = P(I) AND P(I) < = 39 THEN LET A = A + 1
```

```
 80  IF 40 < = P(I) AND P(I) < = 49 THEN LET B = B + 1
 90  IF 50 < = P(I) AND P(I) < = 59 THEN LET C = C + 1
100  IF 60 < = P(I) AND P(I) < = 69 THEN LET D = D + 1
110  NEXT I
120  PRINT "ENTRE 30 y 39 AÑOS", A
130  PRINT "ENTRE 40 y 49 AÑOS", B
140  PRINT "ENTRE 50 y 59 AÑOS", C
150  PRINT "ENTRE 60 Y 69 AÑOS", D
160  END
```

8.7. Sea una lista de veinte elementos, los cuales pueden ser valores positivos, negativos o nulos. Haz un programa que imprima el total de elementos positivos, el total de negativos y el total de nulos.

En este programa se utilizan las variables P, N y O para acumular el número de valores positivos, negativos y nulos, respectivamente.

```
 10  DIM A(20)
 20  LET P = 0 : LET N = 0 : LET O = 0
 30  FOR I = 1 TO 20
 40  INPUT A(I)
 50  IF A(I) > 0 THEN LET P = P + 1
 60  IF A(I) < 0 THEN LET N = N + 1
 70  IF A(I) = 0 THEN LET O = O + 1
 80  NEXT I
 90  PRINT "POSITIVOS", P
100  PRINT "NEGATIVOS", N
110  PRINT "NULOS", O
120  END
```

9 Tablas numéricas

Programas resueltos con DIM A(N,M)

RESUMEN DE LAS INSTRUCCIONES

DIM A(N, M)

Esta instrucción reserva posiciones de memoria en las que se almacenarán los datos de los elementos de la tabla A.

Las posiciones de memoria se nombran con los nombres A(1, 1), A(1, 2), ..., A(2, 1), A(2, 2), etc.

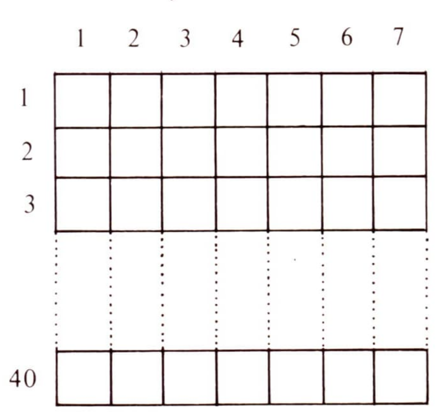


Ejemplo

La instrucción

```
10  DIM A(40, 7)
```

reserva 40 × 7 = 280 posiciones de memoria para la variable de dos índices A. En este caso, la tabla tiene 40 filas y 7 columnas.

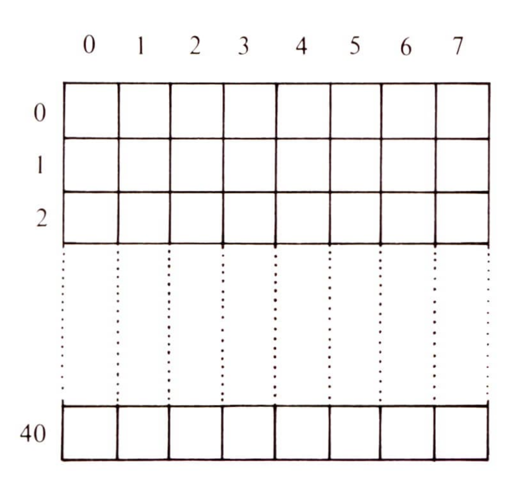


Observación

Hay microordenadores que comienzan a contar desde cero (0), luego la instrucción

10 DIM A(40, 7)

reservaría, es este caso, 41 × 8 = 328 posiciones de memoria para la variable de dos índices A. La tabla tendría, entonces, 41 filas (0, 1, 2, ..., 40) y 8 columnas (0, 1, 2, ..., 7).



PROGRAMAS RESUELTOS EXPLICADOS

9.1. Escribe la tabla que imprimiría en la pantalla el siguiente programa:

```
  5 REM TABLA
 10 DIM A (2,3)
 20 FOR I = 1 TO 2
 30 FOR J = 1 TO 3
 40 READ A(I,J)
 50 NEXT J
 60 NEXT I
 70 DATA −1, 0, 0, 3, 0.5, 8
 80 FOR I = 1 TO 2
 90 FOR J = 1 TO 3
100 PRINT A (I,J)
110 NEXT J
120 NEXT I
130 END
```

— En los dos bucles anidados comprendidos en las líneas **20 - 60** se leen los valores de **A (I,J)** cuyos datos se encuentran en la instrucción **70 DATA.**

— Mediante los dos bucles anidados comprendidos en las líneas **80 - 120** se imprimen los valores de la tabla **A (I,J)** verticalmente. La instrucción que imprime dichos datos es la **100.**

— El resultado que se obtiene en la pantalla es éste:

```
–1
 0
 0
 3
 0.5
 8
```

aunque su distribución en filas y columnas sea la que muestra la tabla

–1	0	0
3	0.5	8

9.2. Haz un programa que almacene en la memoria una tabla de cuatro filas y 10 columnas y a continuación la escriba en la pantalla.

```
 10  DIM A (4,10)
 20  FOR I = 1 TO 4
 30  FOR J = 1 TO 10
 40  INPUT A (I,J)
 50  NEXT J
 60  NEXT I
 70  FOR I = 1 TO 4
 80  FOR J = 1 TO 10
 90  PRINT A (I,J)
100  NEXT J
110  NEXT I
120  END
```

— La instrucción **10** reserva 4 × 10 = 40 posiciones de memoria para la tabla A (I,J).

— Los dos bucles **20 - 60** permiten que se introduzcan los valores de la tabla por medio de la instrucción **40.**

— Los dos bucles **70 - 110** consiguen que la instrucción **90** escriba en la pantalla los valores de la tabla.

9.3. Haz un programa similar al anterior que calcule el promedio de las notas por alumno en todas sus asignaturas.

Si A es el número de alumnos y X el número de asignaturas, la tabla N (A,X) recoge las notas de los A alumnos en las X asignaturas.

El programa es:

```
 10 REM PROMEDIO POR ALUMNO
 20 READ A
 30 READ X
 40 DIM N (A,X)
 50 REM ALMACENAR EN MEMORIA LAS NOTAS
 60 FOR J = 1 TO X
 70 FOR I = 1 TO A
 80 READ N (I,J)
 90 NEXT I
100 NEXT J
110 REM CALCULO DEL PROMEDIO DE CADA ALUMNO
120 FOR I = 1 TO A
130 LET T = 0
140 FOR J = 1 TO X
150 LET T = T + N (I,J)
160 NEXT J
170 PRINT "EL PROMEDIO DEL ALUMNO"; I; "ES"; T/X
180 PRINT
190 NEXT I
300 END
```

— A este programa hay que añadirle instrucciones **DATA** con los datos necesarios. Ejemplo:

```
200 DATA 5, 3
210 DATA 4, 4, 5, 6, 2, 8, 3, 7, 5, 4, 4, 10, 2, 9, 8
```

— En este caso, la instrucción **20** asignará a A el valor 5 (alumnos) y a X el valor 3 (asignaturas).

— La instrucción **40** reservará 5 × 3 = 15 posiciones de memoria para la tabla N (I,J) que contendrá las notas.

— El bucle **60 - 100** permite leer las notas por asignaturas:
4, 4, 5, 6, 2 corresponden a la primera asignatura,
8, 3, 7, 5, 4 corresponden a la segunda asignatura y
4, 10, 2, 9, 8 corresponden a la tercera asignatura.

— En el bucle **120 - 190** se calcula e imprime el promedio de cada alumno.

9.4. Los precios de cinco productos en tres ciudades españolas están registrados en la tabla. Hacer un programa que dé los precios medios de los distintos productos.

	Valencia	Las Palmas	Soria
Naranjas	100	140	120
Plátanos	100	90	130
Fresones	90	150	110
Manzanas	80	100	90
Peras	60	90	90

```
10  DIM A (5, 3)
20  FOR I = 1 TO 5
30  FOR J = 1 TO 3
40  INPUT A(I,J)
50  NEXT J
60  NEXT I
70  FOR I = 1 TO 5
80  LET S = 0
90  FOR J = 1 TO 3
100 LET S = S + A (I,J)
110 NEXT J
120 PRINT S/3
130 NEXT I
140 END
```

— En el bloque de instrucciones **20 - 60** se van introduciendo los distintos precios, por filas.

— El bloque **70 - 130** hace que en S se acumulen los precios de un producto, el cual será dividido por 3. Al comenzar con un nuevo producto se hace S = 0.

PROGRAMAS RESUELTOS

9.1. Señala las variables dimensionadas de dos índices correctamente escritas.

a) X (2 + 2)
b) X (5,5)
c) X (A + B,C)
d) X (X(1,2), X(2,1))
e) X (A,A)

a) La variable X (2 + 2) es de un índice, ya que X (2 + 2) = X (4).
b) X (5,5) está correctamente escrita.
c) X (A + B, C) está correctamente escrita.
d) X (X(1,2), X(2,1)) está correctamente escrita.
e) X (A,A) está correctamente escrita.

9.2. A la siguiente tabla se asigna el nombre A.

1	2
3	4
5	6

a) ¿Qué clase de variable es A?
b) ¿Qué valores tienen A (1,1) y A (3,1)?
c) Si X = 2, e Y = 3, ¿cuál es el valor de A (X,Y), A (X + 1, Y − 1) y A (A(1,2), A(2,1) − 1)?

a) A es una variable de dos índices y representa una tabla con tres filas y dos columnas.
b) A(1,1) = 1 ; A (3,1) = 5.
c) Si X = 2, e Y = 3, se tiene:
A (X,Y) = A (2,3), elemento que no existe en la tabla.
A (X + 1, Y − 1) = A (3,2) = 6
A (A(1,2), A(2,1) − 1) = A (2,2) = 4

9.3. a) Escribe un programa que almacene ceros en una variable que represente a una tabla de 10 filas y 5 columnas.
b) Lo mismo que en el apartado a), pero en lugar de almacenar ceros que almacene 1.

a)
```
10 DIM A(10,5)
20 FOR I = 1 TO 10
30 FOR J = 1 TO 5
40 LET A (I,J) = 0
50 NEXT J
60 NEXT I
70 END
```

Si se tiene en cuenta que la instrucción **10 DIM A (10,5)** asigna ceros a todos los elementos de la tabla A, el programa anterior se puede reducir a estas instrucciones:

```
10  DIM A (10,5)
20  END
```

b)
```
10  DIM A (10,5)
20  FOR I = 1 TO 10
30  FOR J = TO 5
40  LET A (I,J) = 1
50  NEXT J
60  NEXT I
70  END
```

9.4. Al hacer una encuesta para anticipar los posibles resultados de unas próximas elecciones, se han hecho las siguientes preguntas:

a) ¿A qué candidato votaría?

1. NOMBRE 1.
2. NOMBRE 2.
3. EN BLANCO.

b) ¿En qué grupo de edad está usted?

1. Menos de treinta años.
2. Treinta años o más.

Escribe un programa que recuente los votos conseguidos por cada candidato y los clasifique por edades. Los datos de los votos se introducirán con instrucciones **READ** y **DATA.**

Sugerencia: Se puede definir una variable de dos índices, A (X, Y), representando X el candidato votado e Y la edad del votante. Así, A (1,1) es un voto conseguido por el candidato 1 emitido por un votante con edad menor de treinta años.

Cada votante tendrá que dar dos números, X e Y, siendo X el que corresponde al candidato:

1. NOMBRE 1.
2. NOMBRE 2.
3. EN BLANCO.

Y siendo Y el que corresponde al grupo de edad que pertenece:

1. Menos de treinta años.
2. Treinta años o más.

La finalización del proceso se señalará con X = 0 e Y = 0, escribiendo estos dos valores en la instrucción **DATA** en último lugar.

El programa es éste:

```
 10 DIN A (3,2)
 20 READ X,Y
 30 IF X = 0 AND Y = 0 THEN GOTO 60
 40 LET A (X, Y) = A (X, Y) +1
 50 GOTO 20
 60 PRINT "NOMBRE 1 Y MENORES DE 30", A (1,1)
 70 PRINT "NOMBRE 1 Y 30 O MAS", A (1,2)
 80 PRINT "NOMBRE 2 Y MENORES DE 30", A (2,1)
 90 PRINT "NOMBRE 2 Y 30 O MAS", A (2,2)
100 PRINT "EN BLANCO Y MENORES DE 30", A (3,1)
110 PRINT "EN BLANCO Y 30 O MAS", A (3,2)
200 END
```

Como datos se pueden poner, por ejemplo, los siguientes:

```
120 DATA 1, 1, 2, 1, 2, 2, 3, 2, 3, 1, 3, 1, 1, 2, 1, 2, 0, 0
```

9.5. Dadas las calificaciones de 10 alumnos en 5 asignaturas, escribe un programa que obtenga la nota media de cada alumno y la media de cada asignatura.

```
 10 DIM C (10, 5)
 20 FOR I = 1 TO 10
 30 FOR J = 1 TO 5
 40 INPUT C (I,J)
 50 NEXT J
 60 NEXT I
 70 REM MEDIA DE CADA ALUMNO
 80 FOR I = 1 TO 10
 90 LET S = 0
100 FOR J = 1 TO 5
110 LET S = S + C (I,J)
120 NEXT J
130 PRINT "LA MEDIA DEL ALUMNO"; I; "ES"; S/5
140 NEXT I
150 REM MEDIA DE CADA ASIGNATURA
160 FOR J = 1 TO 5
170 LET S = 0
180 FOR I = 1 TO 10
190 LET S = S + C (I,J)
200 NEXT I
210 PRINT "LA MEDIA DE LA ASIGNATURA"; J; "ES"; S/10
220 NEXT J
230 END
```

9.6 Hacer un programa que lea una tabla y obtenga el elemento mayor.

```
 10 INPUT "FILAS"; F
 20 INPUT "COLUMNAS"; C
 30 DIM T (F,C)
 40 FOR I = 1 TO F
 50 FOR J = 1 TO C
 60 INPUT T (I,J)
 70 NEXT J
 80 NEXT I
 90 LET M = T (1,1)
100 FOR I = 1 TO F
110 FOR J = 1 TO C
120 IF M < T (I,J) THEN LET M = T (I,J)
130 NEXT J
140 NEXT I
150 PRINT "EL MAYOR ES:", M
160 END
```

9.7. Una red comercial tiene 10 tiendas y cada tienda 4 departamentos. Escribe un programa que lea los valores de las ventas por tienda y departamento e imprima:

1.º El total de ventas por tienda.

2.º El total de ventas por departamento en todas las tiendas.

La variable V (T,D) indica las ventas en un departamento D de una tienda determinada T. Además, A y B (D) representan las ventas por tienda y por departamento en todas las tiendas, respectivamente.

```
 10 DIM V (10,4)
 20 DIM S (4)
 30 FOR T = 1 TO 10
 40 LET A = 0
 50 FOR D = 1 TO 4
 60 INPUT V (T,D)
 70 LET A = A + V (T,D)
 80 LET B (D) = B (D) + V (T,D)
 90 NEXT D
100 PRINT "EL TOTAL DE VENTAS EN LA TIENDA"; T; "ES"; A
110 NEXT T
120 FOR D = 1 TO 4
130 PRINT "LAS VENTAS DEL DEPARTAMENTO"; D; "EN TODAS
    LAS TIENDAS ES"; B (D)
140 NEXT D
150 END
```

9.8. Haz un programa que lea una tabla y ponga a cero los elementos que estén repetidos.

```
10  INPUT "FILAS"; F
20  INPUT "COLUMNAS"; C
30  DIM T (F,C)
40  FOR I = 1 TO F
50  FOR J = 1 TO C
60  INPUT T (I,J)
70  NEXT J
80  NEXT I
90  FOR I = 1 TO F
100 FOR J = 1 TO C
110 LET A = T (I,J)
120 LET R = 0
130 FOR K = I TO F
140 FOR L = J TO C
150 IF A = T (K,L) THEN LET T (K,L) = 0 : LET R = 1
160 NEXT L
170 NEXT K
180 IF R = 1 THEN LET T (I,J) = 0
190 NEXT J
200 NEXT I
210 END
```

Nota: Si se desea igualar a cero sólo las repeticiones, basta con eliminar las instrucciones **120** y **180**, y sustituir la **150** por **150 IF A = T (K,L) THEN LET T (K,L) = 0.**

9.9. Escribe un programa que lea dos tablas de igual dimensión y forme una tercera sumando los elementos que ocupan la misma posición.

```
10  INPUT "FILAS"; F
20  INPUT "COLUMNAS"; C
30  DIM A (F,C) : DIM B (F,C) : DIM S (F,C)
40  REM LECTURA DE LA PRIMERA TABLA
50  FOR I = 1 TO F
60  FOR J = 1 TO C
70  INPUT A (I,J)
80  NEXT J
90  NEXT I
100 REM LECTURA DE LA SEGUNDA TABLA
110 FOR I = 1 TO F
120 FOR J = 1 TO C
130 INPUT B (I,J)
140 NEXT J
```

```
150  NEXT I
160  REM SUMA DE LAS DOS TABLAS
170  FOR I = 1 TO F
180  FOR J = 1 TO C
190  LET S (I,J) = A (I,J) + B (I,J)
200  NEXT J
210  NEXT I
220  REM IMPRESION DE LA TABLA SUMA
230  FOR I = 1 TO F
240  PRINT "FILA"; I
250  FOR J = 1 TO C
260  PRINT S (I,J)
270  NEXT J
280  NEXT I
290  END
```

9.10. Diseña un programa que lea una tabla y obtenga el elemento menor de cada fila y el elemento mayor de cada columna.

```
 10  INPUT "FILAS"; F
 20  INPUT "COLUMNAS"; C
 30  DIM T (F,C)
 40  FOR I = 1 TO F
 50  FOR J = 1 TO C
 60  INPUT T (I,J)
 70  NEXT J
 80  NEXT I
 90  REM MENOR DE CADA FILA
100  FOR I = 1 TO F
110  LET M = T (I, 1)
120  FOR J = 2 TO C
130  IF T (I,J) < M THEN M = T (I,J)
140  NEXT J
150  PRINT "EL MENOR DE LA FILA"; I; "ES"; M
160  NEXT I
170  REM MENOR DE CADA COLUMNA
180  FOR J = 1 TO C
190  LET M = T (1,J)
200  FOR I = 2 TO F
210  IF T (1,J) > M THEN LET M = T (I,J)
220  NEXT I
230  PRINT "EL MAYOR DE LA COLUMNA"; J; "ES"; M
240  NEXT J
250  END
```

9.11. Escribe un programa que lea una tabla e imprima aquellos elementos que sean los menores de su fila y a la vez los mayores de su columna.

```
 10 INPUT "FILAS"; F
 20 INPUT "COLUMNAS"; C
 30 DIM T (F,C)
 40 FOR I = 1 TO F
 50 FOR J = 1 TO C
 60 INPUT T (I,J)
 70 NEXT J
 80 NEXT I
 90 FOR I = 1 TO F
100 LET M = T (I,1) : LET Y = 1
120 FOR J = 2 TO C
130 IF T (I,J) < M THEN LET M = T (I,J) : LET Y = J
140 NEXT J
150 FOR X = 1 TO F
160 IF (T(X, Y) > M) AND (X < > I) THEN GOTO 190
170 NEXT X
180 PRINT "EL MENOR DE LA FILA"; I; "Y A LA VEZ EL MAYOR
    DE LA COLUMNA"; Y; "ES"; M
190 NEXT I
200 END
```

10 Listas y tablas de cadenas

Programas resueltos con DIM A$(N) y DIM A$(N,M)

RESUMEN DE LAS INSTRUCCIONES

DIM A$(N)

Esta instrucción reserva posiciones de memoria, en las que se almacenarán las cadenas correspondientes a los elementos de la lista A$. (En algunos microordenadores, si N es menor o igual que 10, esta instrucción se puede omitir, pues la reserva se hace automáticamente al utilizar los elementos de la lista A$.)



Ejemplo

La instrucción

```
10   DIM A$(N)
```

reserva posiciones de memoria para las cadenas que se asignan a los elementos A$(1), A$(2), ..., A$(20) de la lista A$. Al principio se les asigna la cadena vacía.

Observaciones

— En los microordenadores que comienzan a contar desde cero los elementos de la lista A$, del ejemplo anterior, son A$(0), A$(1), A$(2), ..., A$(20).

— En los microordenadores tipo **ZX Spectrum** hay que especificar en la instrucción **DIM** el número máximo de caracteres que puede tener las cadenas de la lista. Si, por ejemplo, el número máximo de caracteres es 50, la instrucción del ejemplo hay que sustituirla por

```
10   DIM A$(20,50).
```

DIM A$(N,M)

Esta instrucción reserva posiciones de memoria en las que se almacenarán las cadenas correspondientes a los elementos de la tabla A$. (En algunos microordenadores, si N y M son menores o iguales que 10, esta instrucción se puede omitir.)

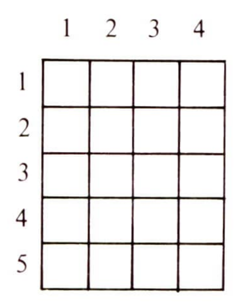


Ejemplo

La instrucción

50 DIM A$ (5,4)

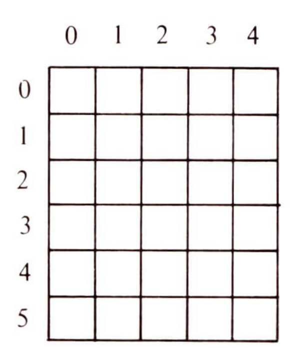
reserva posiciones de memoria para los 5 × 4 = 20 elementos de la variable de dos índices A$. En este caso, la tabla tiene cinco filas y cuatro columnas.



Inicialmente, a los elementos de A$ se les asigna la cadena vacía.

Observaciones

— En los microordenadores que comienzan a contar desde cero, se reservarían 6 × 5 = 30 posiciones de memoria para los 30 elementos de la variable A$, al tener seis filas (0, 1, 2, 3, 4, 5) y cinco columnas (0, 1, 2, 3, 4).



— En los microordenadores tipo **ZX Spectrum** hay que especificar en la instrucción **DIM** el número máximo de caracteres que pueden tener las cadenas de las tablas. Si, por ejemplo, el número máximo de caracteres es como **50,** la instrucción del ejemplo hay que sustituirla por

```
50 DIM A$ (5,4,50)
```

PROGRAMAS RESUELTOS EXPLICADOS

10.1. Explica qué función cumple el siguiente programa:

```
 10 DIM M$(12) (*)
 20 FOR I = 1 TO 12
 30 READ M$(I)
 40 NEXT I
 50 DATA "ENERO", "FEBRERO", "MARZO", "ABRIL"
 60 DATA "MAYO", "JUNIO", "JULIO", "AGOSTO"
 70 DATA "SEPTIEMBRE", "OCTUBRE", "NOVIEMBRE",
    "DICIEMBRE"
100 END
```

Por medio del bucle **20 - 40** y la instrucción **30 READ M$(I),** este programa asigna a los elementos de la lista M$ los meses del año, almacenándolos en la memoria.

10.2. Escribe un programa que almacene en la memoria los días de la semana y los imprima en la pantalla.

```
 10 DIM D$(7) (**)
 20 FOR I = 1 TO 7
 30 READ D$(I)
 40 NEXT I
 50 FOR J = 1 TO 7
 60 PRINT D$(J)
 70 NEXT J
 80 DATA "LUNES", "MARTES", "MIERCOLES", "JUEVES"
 90 DATA "VIERNES", "SABADO", "DOMINGO"
100 END
```

— La instrucción **10** declara una lista D$ de siete elementos.
— El bucle **20 - 40** asigna a los elementos de la lista D$ los días de la semana.
— El bucle **50 - 70** imprime el contenido de la lista D$; es decir, los días de la semana.

(*) Para el ZX Spectrum, cambiar por 10 DIM M$ (12,10).
(**) Para el ZX Spectrum, cambiar por 10 DIM D$ (7,9).

10.1. Escribe un programa que almacene en una variable A$(I) los días de la semana y los imprima junto con su longitud.

(N.º de caracteres)

```
10  DIM A$(7) (*)
20  FOR I = 1 TO 7
30  INPUT "DIA DE LA SEMANA"; A$(I)
40  NEXT I
50  FOR I = 1 TO 7
60  LET N = LEN (A$(I))
70  PRINT A$(I), N
80  NEXT I
90  END
```

10.2. Escribe un programa que almacene en una variable A$(I,J) los meses del año junto con su duración y que imprima la tabla.

```
10  DIM A$(12,2) (**)
20  FOR I = 1 TO 12
30  FOR J = 1 TO 2
40  READ A$(I,J)
50  NEXT J
60  NEXT I
70  REM IMPRESION DE LA TABLA
80  FOR I = 1 TO 12
90  PRINT A$(I,1), A$(I,2)
100 NEXT I
110 DATA "ENERO", "31", "FEBRERO", "28 ó 29", "MARZO",
    "31"
120 DATA "ABRIL", "30", "MAYO", "31", "JUNIO", "30"
130 DATA "JULIO", "31", "AGOSTO", "31", "SEPTIEMBRE",
    "30"
140 DATA "OCTUBRE", "31", "NOVIEMBRE", "30", "DICIEM-
    BRE", "31"
150 END
```

(*) Para el ZX Spectrum, cambiar por 10 DIM A$ (7,9).
(**) Para el ZX Spectrum, cambiar por 10 DIM A$ (12,2).

10.3. Haz un programa que lea varios nombres, forme una lista con todos los que empiezan por A y la imprima.

```
10  PRINT "INTRODUCIR APROXIMADAMENTE Y POR EXCESO
    EL NUMERO DE NOMBRES QUE EMPIEZAN POR A"
20  INPUT N
30  DIM A$(N)
40  LET I = 1
50  INPUT "NOMBRE (PARA TERMINAR INTRODUCIR UN 1)"; X$
60  IF X$ = "1" THEN GOTO 90
70  IF LEFT$ (X$, 1) = "A" THEN LET A$(I) = X$ : LET I = I + 1
80  GOTO 50
90  FOR K = 1 TO I – 1
100 PRINT A$(K)
110 NEXT K
120 END
```

En el ZX Spectrum sustituir las instrucciones **30** y **70** por

```
30 DIM A$(N,50)
```

se supone que ningún nombre tiene más de 50 caracteres.

```
70 IF X$(1 TO 1) = "A" THEN LET A$(I) = X$ : LET I = I + 1
```

10.4. Escribe un programa que lea una lista de nombres y elimine los que están duplicados.

Este ejercicio admite dos interpretaciones:

a) Eliminar el original y los que le dupliquen.
b) Eliminar sólo los que dupliquen a un original.

a)

```
10  INPUT "NUMERO DE NOMBRES"; N
20  DIM A$(N)
30  FOR I = 1 TO N
40  INPUT A$(I)
50  NEXT I
60  FOR I = 1 TO N
70  LET R = 0
80  FOR J = I TO N
90  IF A$(J) = A$(I) THEN LET A$(J) = "  " : LET R = 1
100 NEXT J
110 IF R = 1 THEN LET A$(I) = "  "
120 NEXT I
130 REM IMPRESION DE LOS NOMBRES NO DUPLICADOS
140 FOR I = 1 TO N
150 IF A$(I) < > "  " THEN PRINT A$(I)
160 NEXT I
170 END
```

b) Basta con eliminar las instrucciones **70** y **110,** y sustituir la **90** por la siguiente:

```
90  IF A$(J) = X$ THEN LET A$(J) = " "
```

Nota: En el ZX Spectrum sustituir la instrucción **20** por

```
20  DIM A$(N,50)
```

(se supone que la longitud máxima de los nombres es 50).

10.5. Diseñar un programa que lea dos listas de nombres y forme una tercera con los nombres comunes a ambas.

```
 10  INPUT "NUMERO DE NOMBRES DE LA PRIMERA"; N
 20  INPUT "NUMERO DE NOMBRES DE LA SEGUNDA"; M
 30  DIM A$(N)
 40  DIM B$(M)
 50  IF N > M THEN LET P = M : GOTO 70
 60  LET P = N
 70  DIM C$(P)
 80  REM ENTRADA DE LAS LISTAS
 90  FOR I = 1 TO N
100  INPUT A$(I)
110  NEXT I
120  FOR J = 1 TO M
130  INPUT B$(J)
140  NEXT J
150  REM BUSQUEDA DE LOS NOMBRES COMUNES
160  LET R = 1
170  FOR I = 1 TO N
180  FOR J = 1 TO M
190  IF B$(J) = A$(I) THEN LET C$(R) = A$(I) : LET R = R + 1 :
     GOTO 210
200  NEXT J
210  NEXT I
220  REM IMPRESION DE LA LISTA CON LOS NOMBRES CO-
     MUNES
230  FOR K = 1 TO R - 1
240  PRINT C$ (K)
250  NEXT K
260  END
```

Nota: En el ZX Spectrum sustituir las instrucciones **30, 40** y **70** por

```
30  DIM A$(N,50)
40  DIM B$(M,50)
70  DIM C$(P,50)
```

(se supone que la longitud máxima de los nombres es 50).

10.6. Las siete empresas españolas que registraron mayores ventas en 1982 fueron las siguientes:

Empresa	Ventas (mill. de ptas.)	Beneficios (mill. de ptas.)	Empleados
Campsa	1.094.685	4.890	9.319
Empetrol	522.615	2.734	5.949
Cepsa	381.809	2.136	4.520
Telefónica	266.392	26.546	65.629
Fasa-Renault	184.432	13.877	21.810
Iberia	183.347	–8.124	24.825
Petronor	182.616	2.070	581

Escribe un programa que almacene estos datos en una tabla de caracteres, transforme las ventas, beneficios y número de empleados a valores numéricos y los almacene en una tabla numérica. Por último, deberá imprimir dos columnas, la primera con la relación ventas por empleado (V/E) y la segunda con la relación beneficios por empleado (B/E).

```
10  DIM A$(7.4) (*)
20  DIM B (7, 3)
30  REM ENTRADA DE DATOS POR EMPRESA
40  FOR I = 1 TO 7
50  FOR J = 1 TO 4
60  INPUT A$(I,J)
70  NEXT J
80  NEXT I
90  REM FORMACION DE LA TABLA NUMERICA
100 FOR I = 1 TO 7
110 FOR J = 1 TO 3
120 LET B (I,J) = VAL (A$(I,J + 1))
130 NEXT J
140 NEXT I
150 REM IMPRESION DE LAS RELACIONES V/E Y B/E
160 FOR I = 1 TO 7
170 PRINT B (I,1)/B(I,3), B(I,2)/B(I,3)
180 NEXT I
190 END
```

10.7. Basándote en el ejercicio anterior, diseña un programa que determine en qué empresa la relación ventas por empleado (V/E) es mayor y también la empresa en la que la relación beneficios por empleado es mayor (B/E).

(*) Para el ZX Spectrum, cambiar por 10 DIM A$ (7,4,15).

Basta con sustituir en el progrma del ejercicio anterior el bloque de instrucciones **150 - 190** por el siguiente:

```
150 REM EMPRESA DE MAYOR V/E
160 LET R = 1
170 LET M = B(1,1)/B(1,3)
180 FOR J = 2 TO 7
190 IF B(J,1)/B(J,3) > M THEN LET M = B(J,1)/B(J,3) : LET R = J
200 NEXT J
210 PRINT "LA EMPRESA"; A$(R,1); "ES LA DE MAYOR V/E"; M
220 REM EMPRESA DE MAYOR B/E
230 LET R = 1
240 LET M = B(1,2)/B(1,3)
250 FOR J = 2 TO 7
260 IF B(J,2)/B(J,3) > M THEN LET M = B(J,2)/B(J,3) : LET R = J
270 NEXT J
280 PRINT "LA EMPRESA"; A$(R,1); "ES LA DE MAYOR B/E"; M
290 END
```

11 Funciones

Programas resueltos con SGN, ABS, INT, RND, etc.

RESUMEN DE LAS INSTRUCCIONES

El lenguaje BASIC utiliza varias funciones matemáticas, cuyo nombre, notación y características se transcriben en esta tabla.

Nombre	Notación algebraica	Notación en BASIC	Valor
Signo	sgn x	SGN (X)	+ 1 si X > 0 0 si X = 0 − 1 si X < 0
Valor absoluto	$\|x\|$	ABS (X)	X si X ⩾ 0 − X si X < 0
Raíz cuadrada	$\sqrt{x}$	SQR (X)	X si X ⩾ 0 error si X < 0
Parte entera	E (x)	INT (X)	Parte entera de X (el entero más próximo igual o menor que X)
Logaritmo natural o neperiano	ln x	LOG (X) (LN[X], en ZX Spectrum)	ln X si X > 0 error si X = 0 error si X < 0
Exponencial	e^x	EXP (X)	potencias del número 3, de exponente X (e = = 2,71828)
Seno	sen x	SIN (X)	seno de X (X, ángulo de radianes)

Nombre	Notación algebraica	Notación en BASIC	Valor
Coseno	cos x	COS (X)	coseno de X (X, ángulo de radianes)
Tangente	tg x	TAN (X)	tangente de X (X, ángulo de radianes)
Arco tangente	arc tg x	ATN (X)	arco cuya tangente es X

RND

Esta función genera un número aleatorio entre 0 y 1, excluido este último.



Ejemplo

La instrucción

```
10 LET X = RND
```

asigna a la variable X un número aleatorio comprendido entre 0 y 1, excluido este último.

Observaciones

- Esta función toma formas distintas según los microordenadores. En algunos hay que poner un número entre paréntesis a continuación de RND.

  Ejemplo:

  ```
  10 LET X = RND (1)
  ```

- En los microordenadores MSX hay que utilizar **RND (1)** y además, al principio del programa, se debe poner una asignación del tipo

  ```
  10 Z = RND (–TIME)
  ```

  donde la variable Z puede ser sustituida por cualquier otra.

- En los microordenadores ZX Spectrum hay que utilizar **RND** y añadir al principio del programa la instrucción:

  ```
  10 RANDOMIZE
  ```

DEF FN

Esta instrucción sirve para definir funciones.



Ejemplos

- La instrucción

  10 DEF FN A(X) = X ↑ 5

  define la función FN A de variable X, como la **quinta potencia de X.** (Para definir otra función hay que cambiar la letra A.)

- La instrucción

  10 DEF FN Z$(N) = STR$(N) + "   "

  define la función de cadena FN Z$ de variable N, de acuerdo a este criterio: para todo número N, da como resultado una cadena formada por las cifras de N seguida de cuatro espacios en blanco.

FN A(X)

Da el valor de la función FN A, previamente definida, para el valor X.



Ejemplo

La instrucción

50 Z = FN A(3)

asigna a Z el valor de la función FN A para el valor 3.

Observación

La definición de una función **DEF FN** debe estar antes de su utilización **FN** en el programa.

11.1. Haz un programa que exprese una cantidad de segundos, en horas, minutos y segundos.

```
10 INPUT "SEGUNDOS"; S
20 LET M = INT (S/60)
30 LET S = S - M * 60
40 LET H = INT (M/60)
50 LET M = M - H * 60
60 PRINT H; "HORAS"
70 PRINT M; "MINUTOS"
80 PRINT S; "SEGUNDOS"
90 END
```

— En la línea **10** se introducen los segundos, y en la línea **20** se pasan a minutos, tomando sólo la parte entera.
— La línea **30** determina el número de segundos S sobrantes.
— La línea **40** pasa los minutos a horas, tomando sólo la parte entera.
— La línea **50** determina el número de números M sobrantes.
— Las líneas **60, 70** y **80** imprimen el resultado.

11.2. Haz un programa que exprese en grados, minutos y segundos la amplitud de un ángulo dado en segundos.

Este programa es similar al anterior. Para obtenerlo, basta con sustituir horas por grados.

```
10 INPUT "SEGUNDOS"; S
20 LET M = INT (S/60)
30 LET S = S - M * 60
40 LET G = INT (M/60)
50 LET M = M - G * 60
60 PRINT G; "GRADOS"
70 PRINT M; "MINUTOS"
80 PRINT S; "SEGUNDOS"
90 END
```

— La instrucción **20** pasa los segundos a minutos, tomando su parte entera.
— La instrucción **30** calcula los segundos sobrantes.
— La instrucción **40** pasa los minutos a horas, tomando su parte entera.

— La instrucción **50** calcula los minutos sobrantes.
— Las instrucciones **60, 70** y **80** imprimen los resultados.

11.3. Indica qué escribirá en la pantalla el siguiente programa, si en el momento de su ejecución se introduce 0.451966.

```
10 INPUT "INTRODUCE X"; X
20 LET Y = INT (X * 1000 + 0.5) / 1000
30 PRINT Y
40 END
```

La línea **20** llega al resultado después que el microordenador hace estos cálculos:

X = 0.451966
X * 1000 = 451.966
X * 1000 + 0.5 = 452.466
INT (X * 1000 + 0.5) = 452
INT (X * 1000 + 0.5) / 1000 = 0.452
Y = 0.452

Luego el resultado que aparece en la pantalla es 0.452.

11.4. Realiza los cálculos indicados en las siguientes expresiones para X = −3.1482.

a) INT (X * 10 + 0.5) / 10
b) INT (X * 100 + 0.5) / 100
c) INT (X * 1000 + 0.5) / 1000

a) X * 10 = −31.482
X * 10 + 0.5 = −30.982
INT (X * 10 + 0.5) = −31
INT (X * 10 + 0.5) / 10 = −3.1

b) X * 100 = −314.82
X * 100 + 0.5 = −314.32
INT (X * 100 + 0.5) = −315
INT (X * 100 + 0.5) / 100 = −3.15

c) X * 1000 = −3148.2
X * 1000 + 0.5 = −3147.7
INT (X * 1000 + 0.5) = −3148
INT (X * 1000 + 0.5) / 1000 = −3.148

Observación: Lo que realmente se hace es redondear el número X = −3.1482 a un decimal, a dos decimales y a tres decimales, respectivamente.

11.5. Escribe los programas que permitan redondear:

a) A una cifra decimal.
b) A dos cifras decimales.
c) A n cifras decimales.

a)
```
10 INPUT X
20 LET Y = INT (X * 10 + 0.5) / 10
30 PRINT Y
40 END
```

Si, por ejemplo, X = 3.152, en la línea **20** se tendrá lo siguiente:

X * 10 = 31.52
X * 10 + 0.5 = 32.02
INT (X * 10 + 0.5) = 32
INT (X * 10 + 0.5) / 10 = 3.2

b)
```
10 INPUT X
20 LET Y = INT (X * 100 + 0.5) / 100
30 PRINT Y
40 END
```

Si, por ejemplo, se toma X = 21.342 en la línea **20** se tendrá lo siguiente:

X * 100 = 2134.2
X * 100 + 0.5 = 2134.7
INT (X * 100 + 0.5) = 2134
INT (X * 100 + 0.5) / 100 = 21.34

c) Si para redondear a una cifra decimal hay que multiplicar por 10 y dividir por 10 y para redondear a dos cifras decimales hay que multiplicar por 100 y dividir por 100, parece lógico pensar que para redondear a N cifras decimales hay que multiplicar por 10^N y dividir por 10^N.

```
10 INPUT X
20 INPUT "DECIMALES"; N
30 LET Y = INT (X * 10 ↑ N + 0.5) / 10 ↑ N
40 PRINT Y
50 END
```

11.6. Haz un programa que simule el lanzamiento de una moneda.

```
 10 LET C = 0 : LET Z = 0
 20 INPUT "INTRODUCE N (NUMERO DE LANZAMIENTOS)"; N
 30 FOR I = 1 TO N
 40 LET X = INT (RND(1) * 2 + 1)
 50 IF X = 1 THEN LET C = C + 1
 60 IF X = 2 THEN LET Z = Z + 1
 70 NEXT I
 80 CLS
 90 PRINT : PRINT "EN"; N; "LANZAMIENTOS HA SALIDO:"
100 PRINT : PRINT "*"; C; "VECES CARA"
110 PRINT : PRINT "*"; Z; "VECES CRUZ"
120 END
```

— Las variables C y Z guardan el número de caras y el número de cruces, respectivamente.
— El bucle **30 - 70** simula los N lanzamientos.
— En la línea **40** se obtienen los números aleatorios 1 y 2, que representan cara y cruz.
— Las instrucciones **90, 100** y **110** imprimen los resultados.

Nota: En los microordenadores MSX se debe añadir la instrucción **5 A = RND (–TIME),** y en los microordenadores ZX Spectrum hay que añadir la instrucción **5 RANDOMIZE** y sustituir la línea **40** por **40 LET X = INT (RND * 2 + 1).**

11.7. Indica qué números al azar generan las siguientes instrucciones:

a) X = INT (RND (1) * 9)
b) X = INT (RND (1) * 8 + 1)
c) X = INT (RND (1) * 10 + 11)

a) En este caso, X variará entre $0 * 9 = 0$ y $1 \times 9 = 9$, éste excluido: $(0 \leqslant X < 9)$. Al tomar la parte entera se tiene: X = 0, 1, 2, 3, 4, 5, 6, 7, 8.
b) Los valores de X estarán entre $0 * 8 + 1 = 1$ y $1 * 8 + 1 = 9$, éste excluido: $(1 \leqslant X < 9)$. Al tomar la parte entera, se tiene: X = 1, 2, 3, 4, 5, 6, 7, 8.
c) Los valores de X estarán entre $0 * 10 + 11 = 11$ y $1 * 10 + 11 = 21$, éste excluido: $(11 \leqslant X < 21)$. Al tomar la parte entera se tiene: X = 11, 12, 13, 14, 15, 16, 17, 18, 19, 20.

11.8. Escribe un programa que genere al azar los números 10, 11, 12, 13 y 14.

```
10 LET X = INT (RND (1) * 5 + 10) (*)
20 PRINT X
30 END
```

(*) En el ZX Spectrum, sustituir (RDN (1) por RND.

Los valores de X están comprendidos entre 0 * 5 + 10 = 10 y 1 * 5 + 10 = 15, éste excluido: (10 ⩽ X < 15). Al tomar la parte entera, se obtienen los valores X = 10, 11, 12, 13, 14.

11.9. Indica qué función define el siguiente programa y qué imprime en la pantalla.

```
10 DEF FNR (X) = X ↑ 4
20 FOR X = 1 TO 10
30 PRINT X, FN R(X)
40 NEXT X
50 END
```

Este programa define la función cuarta potencia e imprime en la pantalla una tabla de dos columnas. En la columna izquierda escribe los números del 1 al 10, y en la derecha, sus respectivas potencias cuartas.

```
1     1
2     16
3     81
4     256
5     625
6     1296
7     2401
8     4096
9     6561
10    10000
```

11.10. Escribe un programa que defina una función para calcular los cubos de los veinte primeros números naturales, y después los imprima.

```
10 DEF FN C(X) = X ↑ 3
20 FOR X = 1 TO 20
30 PRINT FIN C(X)
40 NEXT X
50 END
```

— La instrucción **10** define la función **FN C(X)**.
— La instrucción **30** calcula e imprime la función **FN C(X)** para X = 1, 2, 3, ..., 20.

11.11. Haz un programa que defina la función polinómica $3X^2 + 2X + 1$ e imprima una tabla para valores de X comprendidos entre –2 y 2, ambos inclusive, incrementando X en 0.1.

```
10 DEF FN Y(X) = 3 * X ↑ 2 + 2 * X + 1
20 FOR I = −2 TO 2 STEP 0.1
30 PRINT I, FN Y(I)
40 NEXT I
50 END
```

— La instrucción **10** define la función.
— El bucle **20 - 40** imprime la tabla de valores. Observar que los valores que va tomando la variable I se sustituyen en el lugar de X en la instrucción **20.**

11.12. Explica qué hace el siguiente programa:

```
10 DEF FN A(X) = X ↑ 2
20 DEF FN B(X) = X ↑ 3
30 DEF FN S(X) = X ↑ 2 + X ↑ 3
40 FOR X = 1 TO 10
50 PRINT X, FN S(X)
60 NEXT X
70 END
```

— Este programa define tres funciones **FN A(X), FN B(X), FN S(X)** que calculan el cuadrado de X, el cubo de X y la suma del cuadrado y el cubo de X, respectivamente.
— El bucle **40 - 60** imprime una tabla con los valores de X del 1 al 10 y el valor de la función **FN S(X)** para estos valores.

PROGRAMAS RESUELTOS

11.1. Define una función para cada una de las siguientes fórmulas:

a) $Y = \dfrac{X + 5}{2}$

b) $Y = 3x^2 - \dfrac{1}{2}X + 5$

c) $Y = X \cdot 5^X$

d) $Y = X + E(X)$

e) $Y = X + / X / + E(X)$

f) $Y = \dfrac{1}{X^2} + X^2 - \dfrac{X^3 - 1}{5}$

```
a) DEF FN Y(X) = (X + 5) / 2
b) DEF FN Y(X) = 3 * X ↑ 2 −
   −1 / 2 * X + 5
c) DEF FN Y(X) = X * 5 ↑ X
d) DEF FN Y(X) = X + INT(X)
e) DEF FN Y(X) = X + ABS(X) +
   +INT(X)
f) DEF FN Y(X) = 1 / X ↑ 2 +
   +X ↑ 2 − (X ↑ 3 − 1) / 5
```

11.2. Calcula el valor de FN A(6.285) si la función FN A(X) se define así:

```
10  DEF FN A(X) = X - INT(X)
```

FN A(6.285) = 6.285 − INT (6.285) = 6.285 − 6 = 0.285
Se obtiene la parte decimal del número.

11.3. Indica qué precauciones hay que tener a la hora de utilizar las siguientes funciones en un programa:

a) 5 DEF FN B(X) = 1 / (2X − 1)
b) 20 DEF FN C(X) = 5 / (X^2 − 1)

a) En primer lugar, hay que escribir la línea correctamente respetando la sintaxis del lenguaje BASIC.

```
5  DEF FN B(X) = 1 / (2 * X - 1)
```

La precaución que hay que tener es no dar a X valores excesivamente próximos a 0.5, pues, en caso contrario, el denominador sería un valor muy próximo a cero, para el cual la función no tiene valor. Si esto ocurriera se emitiría un mensaje de error.

b) En primer lugar, hay que escribir correctamente la línea **20**:

```
20  DEF FN C(X) = 5 / (X ∧ 2 - 1)
```

La precaución que hay que tener es no dar valores a X muy próximos a 1 y a −1 para evitar que el denominador adquiera valores muy próximos a cero, para los que la función no tiene valores. Se evita así que se emita un mensaje de error.

11.4. Escribe un programa que dé el valor absoluto de un número sin utilizar la función **ABS(X).**

```
10  INPUT X
20  PRINT "EL NUMERO ES:"; X
30  IF X < 0 THEN LET X = -X
40  PRINT "EL VALOR ABSOLUTO ES:"; X
50  END
```

11.5. Cuando se lanzan dos monedas, los resultados posibles son:

+, + que se puede expresar así: 0, 0
c, + que se puede expresar así: 1, 0
+, c que se puede expresar así: 0, 1
c, c que se puede expresar así: 1, 1

Haz un programa que simule el lanzamiento de las dos monedas, e imprima el resultado que se produce.

```
10  LET X = INT (RND(1) * 2)
20  LET Y = INT (RND(1) * 2)
30  IF X = 0 THEN PRINT "+"; : GOTO 50
40  PRINT "C";
50  IF Y = 0 THEN PRINT "+" : GOTO 70
60  PRINT "C"
70  END
```

En el ZX Spectrum, sustituir RND(1) por RND.

11.6. Haz un programa que defina la función polinómica de tercer grado:

$$f(x) = Ax^3 + Bx^2 + Cx + D$$

e introduzca los valores A = 3, B = 2, C = 1 y D = 0.

Evalúa esta función desde I = 1 hasta 10, incrementando a X de 1 en 1.

```
10  DEF FN F(X) = A * X ↑ 3 + B * X ↑ 2 + C * X + D
20  READ A, B, C, D
30  DATA 3, 2, 1, 0
40  FOR I = 1 TO 10
50  PRINT FN F(I)
60  NEXT I
70  END
```

11.7. Elabora un programa que defina la función parte decimal y escriba su valor para cualquier X que se introduzca.

```
10  DEF FN D(X) = X − INT (X)
20  INPUT X
30  PRINT "PARTE DECIMAL:"; FN D(X)
40  END
```

11.8. ¿Cómo se tiene que definir la función parte decimal si el número es negativo?

Si se quiere obtener las cifras decimales se debe definir la función como sigue:

FN D(X) = 1 − (X − INT(X))

Así, por ejemplo, para X = –3.27, se tiene:

INT (X) = –4
N − INT (X) = –3.27 + 4 = 0.73
1 − (X − INT (X)) = 1 − 0.73 = 0.27

11.9. Haz un programa que dé la media de dos números aleatorios, mediante la definición de una función.

```
10  DEF FN A( ) = (RND[1] + RND (1)) / 2
20  PRINT FN A( )
30  END
```

Se puede observar que la función FN A( ) no tiene ningún argumento.

En el ZX Spectrum, sustituir RND (1) por RND.

11.10. Escribe un programa que lea un número y averigüe si es primo o compuesto.

```
10  INPUT N
20  IF N < 2 THEN GOTO 10
30  FOR I = SQR(N) TO 2
40  IF N/I = INT (N/I) THEN PRINT N; "NO ES PRIMO" : GOTO 70
50  NEXT I
60  PRINT N; "ES PRIMO"
70  END
```

11.11. Haz un programa que obtenga la lista de los números primos menores que N.

```
10  INPUT N
20  IF N < 2 THEN GOTO 10
30  FOR K = 2 TO N − 1
40  FOR I = 2 TO SQR (K)
50  IF K/I = INT (K/I) THEN GOTO 80
60  NEXT I
70  PRINT K
80  NEXT K
90  END
```

11.12. Elabora un programa que descomponga un número en sus factores primos.

```
10  INPUT N
20  IF N < 2 THEN GOTO 10
30  FOR I = 2 TO N
40  IF N/I = INT (N/I) THEN PRINT I : LET N = N/I : GOTO 40
50  NEXT I
60  END
```

11.13. Escribe un programa que lea dos números y averigüe si son primos entre sí.

```
10  INPUT N, M
20  LET X = N/M
30  LET Y = M/N
40  IF (X = INT(X)) OR (Y = INT(Y)) THEN PRINT N; "Y"; M;
    "NO SON PRIMOS ENTRE SI" : GOTO 60
50  PRINT N; "Y"; M; "SON PRIMOS ENTRE SI"
60  END
```

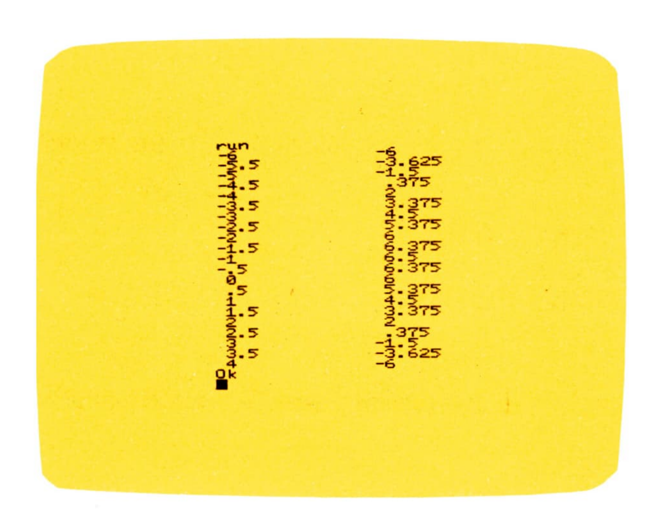
11.14. Realiza un programa que lea dos números y averigüe si tienen factores primos comunes.

```
10 INPUT N, M
20 IF N < 2 OR M < 2 THEN GOTO 10
30 FOR K = 2 TO N
40 FOR I = 2 TO SQR(K)
50 IF K/I = INT (K/I) THEN GOTO 80
60 NEXT I
70 IF M/K = INT (M/K) THEN PRINT K
80 NEXT K
90 END
```

11.15. Elabora un programa que defina la función $f(x) = -0.5X^2 - X + 6$, calcule una tabla de valores incrementando a X de 0.5 en 0.5 e imprima la tabla.

```
10 DEF FN F(X) = –0.5 * X ↑ 2 – X + 6
20 FOR X = –6 TO 4 STEP 0.5
30 PRINT X, FN F(X)
40 NEXT X
50 END
```

Los valores de X se han tomado entre –6 y 4, pues el vértice de la parábola que representa a la función tiene por abscisa X = –1.



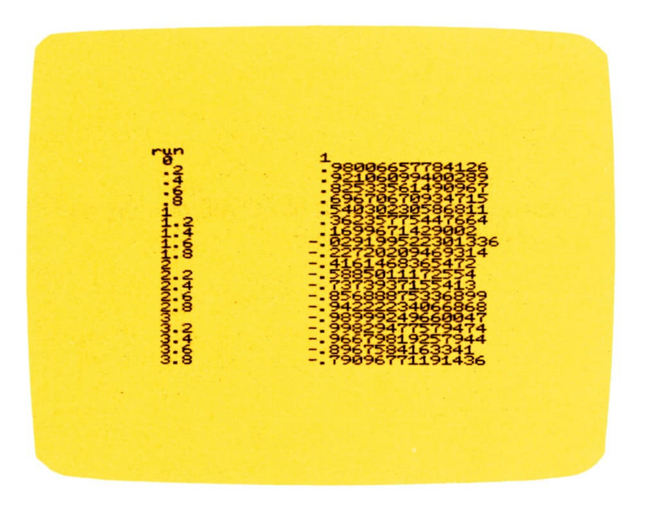
11.16. Escribe un programa que genere e imprima una tabla de valores para la función f (x) = sen X, entre X = 0 y X = 6.28, e incrementando a X de 0.2 en 0.2. (Con la tabla obtenida, representa esta función en papel milimetrado.)

```
10 FOR X = 0 TO 6.28 STEP 0.2
20 PRINT X, SIN(X)
30 NEXT X
40 END
```



11.17. Haz lo mismo que en el ejercicio anterior con la función f (x) = cos X.

```
10 FOR X = 0 TO 6.28 STEP 0.2
20 PRINT X, COS(X)
30 NEXT X
40 END
```



12 Subrutinas

Programas resueltos con GOSUB - RETURN, ON ... GOSUB, ON ... GOTO

RESUMEN DE LAS INSTRUCCIONES

GOSUB

Esta instrucción transfiere el control a la instrucción en la que comienza la subrutina.



Ejemplo

Cuando se ejecuta la instrucción

```
30   GOSUB 1000
```

el microordenador envía el control a la instrucción **1000**, para que se ejecute la subrutina que comienza en esa línea.

RETURN

Devuelve el control a la instrucción siguiente a la GOSUB que previamente había transferido el control a la subrutina.



Ejemplo

Cuando se ejecuta la línea

```
1200   RETURN
```

el microordenador devuelve el control a la línea siguiente a la **GOSUB**, que había llamado previamente a la subrutina.

ON ... GOSUB

Esta instrucción transfiere el control a una o a varias subrutinas.



Ejemplo

La instrucción

```
50   ON V GOSUB 1000, 2000, 3000, 500
```

envía el control a las subrutinas **1000, 2000, 3000** ó **500**, según que V valga 1, 2, 3 ó 4, respectivamente.

ON ... GOTO

Esta instrucción transfiere el control a una o a varias líneas.



Ejemplo

La instrucción

```
70   ON V GOTO 100, 50, 400
```

envía el control a las instrucciones **100, 50** ó **400,** según que V valga 1, 2 ó 3, respectivamente.

PROGRAMAS RESUELTOS EXPLICADOS

12.1. Explica cómo funciona el siguiente programa al ser ejecutado:

```
10  INPUT "A="; A
20  INPUT "B="; B
30  GOSUB 200
40  GOSUB 300
50  PRINT "P="; P; "C="; C
60  GOTO 10
200 REM PRODUCTO
210 LET P = A * B
220 RETURN
300 REM COCIENTE
310 LET C = A/B
320 RETURN
400 END
```

— En este programa se pueden distinguir tres partes: el programa principal (líneas **10-60**), la subrutina **producto** (líneas **200-220**) y la subrutina **cociente** (líneas **300-320**).
— Las líneas **10** y **20** piden valores para A y B.
— La línea **30** llama a la subrutina producto, en la cual se calcula el producto P de A y B.
— La línea **40** llama a la subrutina cociente, en la cual se calcula el cociente C de A y B.
— La línea **50** imprime los resultados.
— La línea **60** envía el control a la línea **10** para que se vuelva a ejecutar el programa.

12.2. Indica lo que ocurrirá al ejecutarse el siguiente programa:

```
10  READ A
20  GOSUB 100
30  PRINT A, B, C
100 REM SUBRUTINA
110 LET B = A ↑ 2
120 LET C = A ↑ 3
200 DATA –3, 2, 1, 8, 4
210 GOTO 10
220 END
```

— En primer lugar hay que advertir que este programa tiene un error: un **GOSUB** sin **RETURN.**
— Su ejecución comienza en la línea **10,** asignando a la variable A el valor –3.
— La instrucción **20** transfiere el control a la **100.**

— Las líneas **110** y **120** calculan el cuadrado y el cubo de A = −3. (En el ZX Spectrum se produciría un error al no poder calcular potencias de números negativos.)
— La línea **210** transfiere el control a la línea **10** y se vuelve a repetir el proceso anterior para A = 2.
— Cuando se termine el proceso para todos los valores de A se producirá un error por falta de datos.

12.3. Escribe un programa con dos subrutinas, una para sumar dos números introducidos por el teclado y la otra, para restarlos.

```
 10 INPUT A, B
 20 GOSUB 100
 30 GOSUB 200
 40 END
100 REM SUMA
110 LET S = A + B
120 PRINT "SUMA ="; S
130 RETURN
200 REM RESTA
210 LET R = A - B
220 PRINT "RESTA ="; R
230 RETURN
```

— Este programa consta de tres partes: el programa principal (líneas **10-40**), la subrutina **suma** (líneas **100-130**) y la subrutina **resta** (líneas **200-230**).
— Desde el programa principal se llama a las subrutinas para que realicen las tareas de suma y resta.

12.4. Haz un programa que genere un número aleatorio entero comprendido entre 0 y 10, éste excluido, y que permita introducir un número comprendido también entre 0 y 10.

Además, el programa averiguará mediante una subrutina si el número generado coincide con el número introducido.

```
 10 LET X = INT (RND(1) * 10)
 20 INPUT "ENTERO ENTRE 0 Y 10"; N
 30 GOSUB 100
 40 END
100 REM SUBRUTINA
110 IF X = N THEN PRINT "SI COINCIDEN" : GOTO 130
120 PRINT "NO COINCIDEN"
130 RETURN
```

— En la línea **10** se calcula el número aleatorio X.
— En la línea **20** se introduce el número N.
— La instrucción **30** llama a la subrutina **100,** que compara estos números.

En el ZX Spectrum, sustituir RND(1) por RND.

12.1. Completa el programa siguiente con las instrucciones necesarias para que se ejecute correctamente.

```
 10  INPUT "A ="; A
 20  INPUT "B ="; B
 30  PRINT "A ="; A; "B ="; B, "P ="; P
 40  GOTO 10
100  REM PRODUCTO
110  LET P = A * B
120  RETURN
200  END
```

Hay que añadir una instrucción antes de la línea **30**, que llame a la subrutina

```
25  GOSUB 100
```

12.2. Haz un programa que permita introducir dos números, calcule mediante cuatro subrutinas la suma, resta, multiplicación y división de dichos números, y después los imprima en una misma línea.

```
 10  INPUT X, Y
 20  GOSUB 100
 30  GOSUB 200
 40  GOSUB 300
 50  GOSUB 400
 60  PRINT "SUMA ="; S; "RESTA ="; R; "MULTIPLICACION =";
     M; "DIVISION ="; D
 70  END
100  REM SUMA
110  LET S = X + Y
120  RETURN
200  REM RESTA
210  LET R = X - Y
220  RETURN
300  REM MULTIPLICACION
310  LET M = X * Y
320  RETURN
400  REM DIVISION
410  LET D = X/Y
420  RETURN
```

12.3. Repite el programa anterior utilizando las instrucciones **READ** y **DATA**. Basta con sustituir la instrucción **10** por

```
10 READ X, Y
```

y añadir una instrucción de datos, por ejemplo, **DATA 7, 3.**

```
 10 READ X, Y
 20 GOSUB 100
 30 GOSUB 200
 40 GOSUB 300
 50 GOSUB 400
 60 PRINT "SUMA ="; S; "RESTA ="; R; "MULTIPLICACION =";
    M; "DIVISION ="; D
 70 DATA 7, 3
 80 END
100 REM SUMA
110 LET S = X + Y
120 RETURN
200 REM RESTA
210 LET R = X - Y
220 RETURN
300 REM MULTIPLICACION
310 LET M = X * Y
320 RETURN
400 REM DIVISION
410 LET D = X/Y
420 RETURN
```

12.4. Escribe lo que imprime el siguiente programa al ser ejecutado.

```
 10 READ A, B, C
 20 GOSUB 100
 30 GOSUB 200
 40 GOSUB 300
 50 GOTO 400
 60 DATA 1, 2, 3
100 REM SUBRUTINA 1
110 PRINT A; B; C
120 RETURN
200 REM SUBRUTINA 2
210 PRINT A ↑ 2; B ↑ 2; C ↑ 2
220 RETURN
300 REM SUBRUTINA 3
310 PRINT A ↑ 3; B ↑ 3; C ↑ 3
320 RETURN
400 END
```

Imprimirá los valores de A, B y C, sus cuadrados y sus cubos.

```
run
1    2    3
1    4    9
1    8    27
```

12.5. Elabora un programa que lea cuatro números positivos y mediante tres subrutinas imprima en tres líneas:

— los cuatro números leídos,
— su respectiva parte entera y
— su respectiva parte decimal.

```
10  INPUT A, B, C, D
20  GOSUB 100
30  GOSUB 200
40  GOSUB 300
50  END
100 REM SUBRUTINA 1
110 PRINT A; B; C; D
120 RETURN
200 REM SUBRUTINA 2
210 PRINT INT (A); INT (B); INT (C); INT (D)
220 RETURN
300 REM SUBRUTINA 3
310 PRINT A - INT (A); B - INT (B); C - INT (C); D - INT (D)
320 RETURN
```

12.6. Haz un programa que genere dos números aleatorios enteros A y B del 1 al 6, ambos inclusive, y mediante una subrutina averigüe cuál de ellos es el mayor, dando un mensaje al respecto.

El programa permitirá repetir el proceso todas las veces que se desee, y la ejecución del mismo deberá interrumpirse entre dos procesos sucesivos.

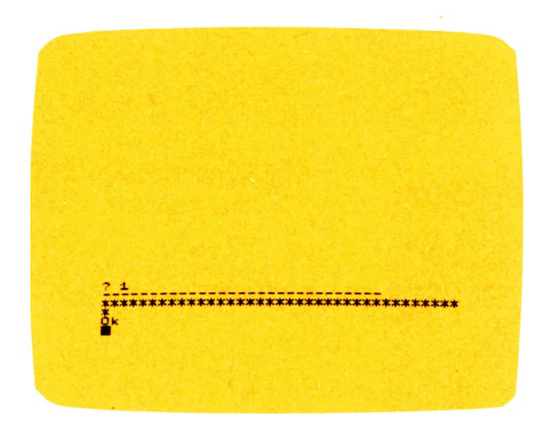
```
10  LET X = INT (RND(1) * 6 + 1)
20  LET Y = INT (RND(1) * 6 + 1)
30  GOSUB 100
40  INPUT "PARA REPETIR EL PROCESO PULSAR 1"; N
```

```
50  IF N = 1 THEN GOTO 10
60  END
100 REM SUBRUTINA
110 IF X > Y THEN PRINT X; "ES MAYOR QUE"; Y
120 IF X < Y THEN PRINT Y: "ES MAYOR QUE"; X
130 IF X = Y THEN PRINT X; "ES IGUAL QUE"; Y
140 RETURN
```

12.7. Dibuja lo que imprimiría el siguiente programa al ser ejecutado, si se introduce el valor 1 para A.

```
1   CLS
10  INPUT A
20  ON A GOSUB 100, 200
30  IF A > 2 THEN GOTO 300
40  LET A = A + 1
50  GOTO 20
100 REM SUBRUTINA 1
110 PRINT "..........................................................................  "
120 RETURN
200 REM SUBRUTINA 2
210 PRINT "*********************************"
220 RETURN
300 END
```

Al ejecutarlo para A = 1 imprimirá una línea de guiones por medio de la subrutina 1.



12.8. ¿Qué imprimirá el programa anterior si se introduce el valor 2 para A? ¿Y si se introduce el valor 3?

Para A = 2 imprimirá una fila de asteriscos. Y para A = 3 no imprimirá nada.

12.9. Cambia la instrucción **20** del programa escrito en el ejercicio 12.7, por **20 ON A GOTO 100, 200** y analiza lo que aparecería en la pantalla al ser ejecutado, con los valores 1, 2 y 3 para A.

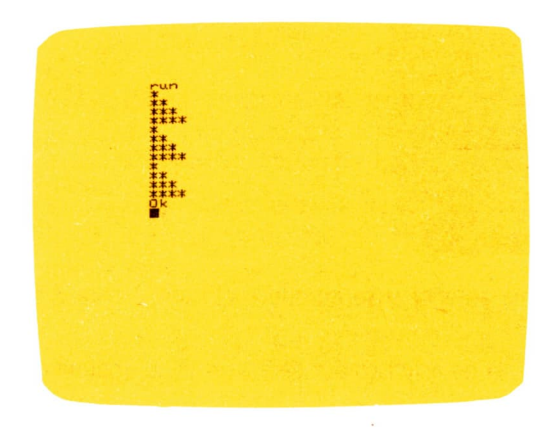
— Si A = 1, la instrucción **20** envía el control a la línea **100** imprimiéndose una fila de opciones y un mensaje de error por no encontrarse antes de la instrucción **120 RETURN** una instrucción **GOSUB** que transfiera el control a la subrutina.

— Si A = 2, la instrucción **20** envía el control a la línea **200,** se imprime una fila de asteriscos y se produce un error por no encontrarse antes de la instrucción **220 RETURN** una instrucción **GOSUB** que transfiere el control a la subrutina.

— Si A = 3, el programa se ejecutaría sin imprimir nada.

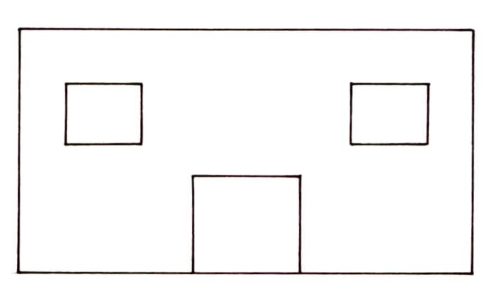
12.10. El siguiente programa contiene una subrutina. Dibuja lo que imprime en pantalla al ser ejecutado.

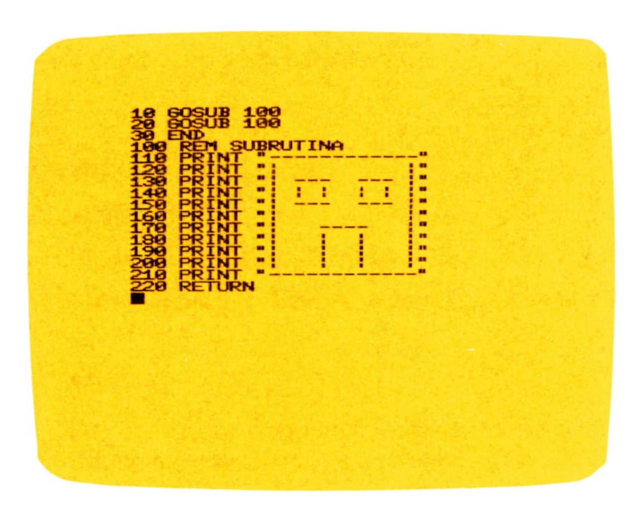
```
 10 LET PROCEDIMIENTO = 200
 20 FOR I = 1 TO 3
 30 GOSUB PROCEDIMIENTO
 40 NEXT I
 50 GOTO 300
200 REM PROCEDIMIENTO
210 PRINT "*"
220 PRINT "**"
230 PRINT "***"
240 PRINT "****"
250 RETURN 40
300 END
```



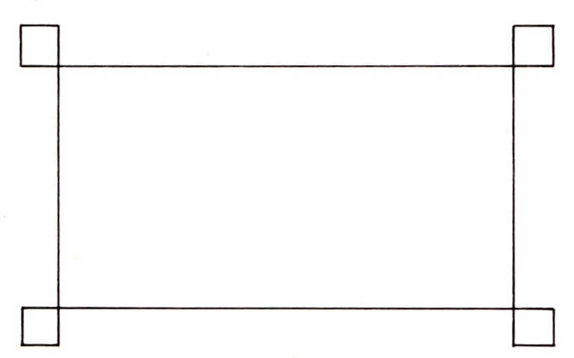
En los microordenadores del tipo MSX, eliminar la línea **10** y sustituir la **30** por **30 GOSUB 200,** pues no admiten variables en un **GOSUB.**

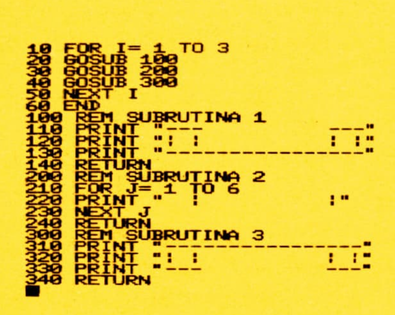
12.11. Haz un programa con una subrutina que dibuje dos veces en la pantalla la siguiente figura:





12.12. Elabora un programa con tres subrutinas que dibuje tres veces la siguiente figura:





10 FOR I= 1 TO 3
20 GOSUB 100
30 GOSUB 200
40 GOSUB 300
50 NEXT I
60 END
100 REM SUBRUTINA 1
110 PRINT "---   ---"
120 PRINT ": :   : :"
130 PRINT "--------------------"
140 RETURN
200 REM SUBRUTINA 2
210 FOR J= 1 TO 6
220 PRINT " :   :"
230 NEXT J
240 RETURN
300 REM SUBRUTINA 3
310 PRINT "--------------------"
320 PRINT ": :   : :"
330 PRINT "---   ---"
340 RETURN

13 Impresión ordenada

Programas resueltos con PRINT TAB, PRINT AT y LOCATE

RESUMEN DE LAS INSTRUCCIONES

PRINT TAB

Esta instrucción imprime a partir de la columna que se especifique a continuación de la palabra TAB



Ejemplo

Al ejecutarse la instrucción

```
10  PRINT TAB (15); "TERESA"
```

se imprime la cadena "TERESA" a partir de la columna 15.

PRINT AT

Esta instrucción imprime a partir de la fila y la columna que se especifique a continuación de la palabra AT.



Ejemplo

Al ejecutarse la instrucción

```
20  PRINT AT 7, 10; X
```

se imprime el valor de la variable X a partir de la posición determinada por la intersección de la fila 7 y la columna 10.

Observación

La instrucción **PRINT AT** la poseen los microordenadores del tipo ZX Spectrum. Sin embargo, la mayoría de los microordenadores tiene una instrucción que realiza la misma función, aunque con distinto formato (consultar manual).

LOCATE

Esta instrucción sitúa el cursor en la columna y en la fila que se especifique a continuación de la palabra LOCATE.



Ejemplos

- La instrucción

  50 LOCATE 10, 5

  sitúa el cursor en la posición determinada por la intersección de la columna 10 y la fila 5.
- La instrucción

  60 LOCATE 10

  sitúa el cursor en la columna 10.
- La instrucción

  70 LOCATE, 5

  sitúa el cursor en la fila 5.

PROGRAMAS RESUELTOS EXPLICADOS

13.1. Haz un programa que escriba como cabecera "NOMBRE" a partir de la columna 10, y "APELLIDO", a partir de la 20. Además, permitirá introducir nombres y apellidos de personas, imprimiéndolos a partir de la columna 10 y 20, respectivamente.

```
10 PRINT TAB (10); "NOMBRE"; TABB (20); "PRIMER APELLIDO"
20 INPUT N$
30 INPUT A$
40 PRINT TAB (10); N$; TAB (20); A$
50 INPUT "HAY MAS NOMBRES"; X$
60 IF X$ = "SI" THEN GOTO 20
70 END
```

— La línea **10** imprime la cabecera y la **40**, los nombres y apellidos.

13.2. ¿Qué aparecerá en la pantalla al ejecutar el siguiente programa? ¿Cada vez que se ejecute aparecerá el mismo contenido en la pantalla?

```
10 RANDOMIZE
20 FOR I = 1 TO 10
30 LET N = INT (10 * RND)
40 PRINT TAB (N); "BUENOS DIAS"; TAB (25); N
50 NEXT I
60 END
```

— Aparecerá diez veces la frase "BUENOS DIAS" y el número de la columna en que comienza a imprimirse cada vez.
— Cada vez que se ejecute el programa, la impresión en pantalla variará de posición debido a las instrucciones **RANDOMIZE** y **RND**.

Nota: En los microordenadores MSX hay que sustituir la instrucción **10** por **10 LET A = RND (–TIME),** y en la **30,** en lugar de **RND** hay que utilizar **RND**(1).

13.3. Haz un programa que imprima una A en cada una de las 22 filas, de tal manera que en cada fila la posición de la columna se obtenga de forma aleatoria, entre las 32 posibles.

```
10 FOR I = 0 TO 21
20 LET X = INT (RND (1) * 32)
30 PRINT TAB (X); "A"
40 NEXT I
50 END
```

En la línea **20** se calcula el valor aleatorio X de la columna, que varía de 0 a 31.

Nota: En los microordenadores ZX Spectrum hay que sustituir RND(1) por RND.

13.4. Haz un programa que dibuje un asterisco en la posición determinada por la intersección de la fila 11 y columna 16. (Para llegar a la fila 11, introducir un bucle.)

```
10 FOR I = 0 TO 10
20 PRINT
30 NEXT I
40 PRINT TAB (16); "*"
50 END
```

— El bucle **10 - 30** imprime once filas en blanco (de la 0 a la 10); así se llega a la fila 11.
— La instrucción **40** imprime un asterisco en la posición determinada por la intersección de la columna 16.

13.5. Realiza un programa que llene la columna 12 de asteriscos.

```
10 FOR I = 0 TO 21
20 PRINT TAB (12); "*"
30 NEXT I
40 END
```

La línea **20** imprime el asterisco en la columna 12.

Esta operación la repite en las filas 0 a 21 por medio del bucle **10 - 30.**

13.6. Haz un programa que imprima A en toda la fila 13.

```
10 FOR I = 0 TO 12
20 PRINT
30 NEXT I
40 FOR J = 0 TO 31
50 PRINT "A";
60 NEXT J
70 END
```

— El bucle **10 - 30** baja el cursor a la fila 13.
— El bucle **40 - 60** imprime A en toda la fila 13.

13.7. Escribe un programa que dibuje una recta que vaya de (0,0) a (21,21).

Los puntos pertenecientes a la recta se indican con asteriscos para que en la pantalla se visualicen mejor.

```
10 FOR I = 0 TO 21
20 PRINT TAB (I); "*"
30 NEXT I
40 END
```

Cada vez que se imprime un asterisco se baja una fila y se desplaza una posición a la derecha del cursor (incremento de I).

13.8. Haz un programa que dibuje una recta de (0,31) a (21,10).

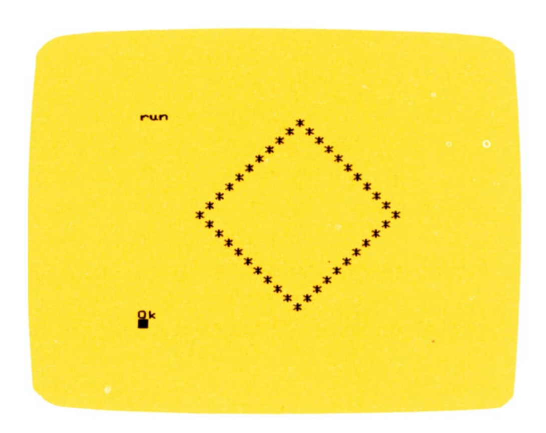
```
10 FOR I = 0 TO 21
20 PRINT TAB (31 – I); "*"
30 NEXT I
40 END
```

— Las filas varían de 0 a 21, según aumenta la variable I del bucle.
— Las columnas varían de 31 – 0 = 31 a 31 – 21 = 10, en función de la variación de 31 – I.

13.9. ¿Qué figura geométrica dibuja este programa?

```
10 PRINT TAB (16); "*"
20 FOR I = 1 TO 10
30 PRINT TAB (16 – I); "*"; TAB (16 + I); "*"
40 NEXT I
50 FOR I = 9 TO 1 STEP –1
60 PRINT TAB (16 – I); "*"; TAB (16 + I); "*"
70 NEXT I
80 PRINT TAB (16); "*"
90 END
```

— La línea **10** imprime un asterisco en la columna 16.
— El bucle **20 - 40** imprime dos asteriscos en cada una de las filas que van de 1 a 10. Cada par de asteriscos son simétricos respecto de la columna 16, y cada vez aumenta su distancia en dos posiciones.
— El bucle **50 - 70** imprime dos asteriscos en cada una de las nueve filas siguientes. Cada par de asteriscos son simétricos respecto de la columna 16, y cada vez disminuye su distancia en dos posiciones.
— La línea **80** imprime un asterisco en la columna 16.
— En conclusión, la figura que dibuja es un rombo.

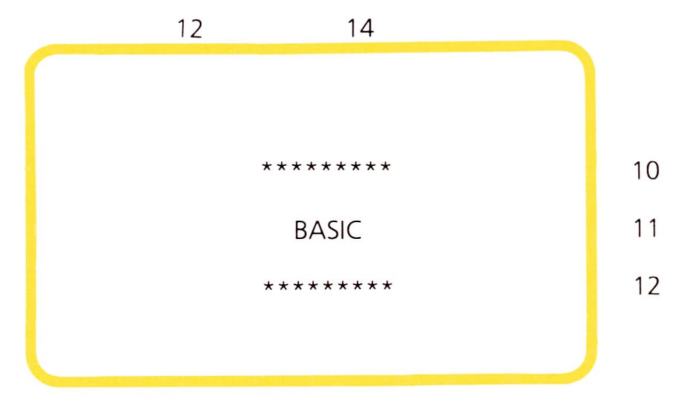


13.10. Indica qué escribe el siguiente programa, y en qué posición de la pantalla.

```
 5 REM ESCRIBE EN LA PANTALLA
10 PRINT
20 PRINT AT 10, 12; "*********"
30 PRINT AT 11, 14; "BASIC"
40 PRINT AT 12, 12; "*********"
50 PRINT
60 END
```

En los equipos MSX la instrucción **PRINT AT** se sustituye por **LOCATE**

— Escribe la palabra BASIC en la posición determinada por la intersección de la fila 11 y la columna 14.
— En las filas superior e inferior (10 y 12) imprime nueve asteriscos a partir de la columna 12.



13.11. Haz un programa que lea un nombre y lo imprima 10 veces; la primera vez a partir de la fila 1 y columna 1, la segunda a partir de la fila 2 y columna 2, y así sucesivamente.

```
10 INPUT A$
20 CLS
30 FOR I = 1 TO 10
40 PRINT AT I, I; A$
50 NEXT I
60 END
```

— La línea **10** permite introducir el nombre, y la **20** borra la pantalla.
— El bucle **30 - 50** imprime el nombre 10 veces a partir de la fila 1 y columna 1, a partir de la fila 2 y columna 2, ..., a partir de la fila 10 y columna 10.

Nota: En los microordenadores MSX, sustituir la línea **40** por **40 LOCATE I, I : PRINT A$.**

13.12. Escribe un programa tal que al introducir una frase calcule su longitud L y la imprima a partir de la fila L y columna L.

```
10 INPUT A$
20 CLS
30 LET L = LEN (A$)
40 PRINT AT L, L; A$
50 END
```

— La línea **10** permite introducir la frase, y la **20** borra la pantalla.
— La línea **30** calcula la longitud L de la frase y la **40** la imprime en la fila L, columna L.
— La longitud L debe ser inferior al número de filas y al número de columnas para que no se produzca error.

Nota: En los microordenadores MSX, sustituir la línea **40** por **40 LOCATE L, L : PRINT A$.**

13.13. Haz un programa que dibuje una línea recta de asteriscos en la fila 3.

```
10  FOR I = 0 TO 31
20  PRINT AT 3, I; "*"
30  NEXT I
40  END
```

El bucle **10 - 30** imprime la recta de asteriscos dejando fija la fila 3 y variando el número de columna I (instrucción **20**).

Nota: En los microordenadores MSX hay que sustituir la línea **20** por **20 LOCATE I, 3 : PRINT "*".**

13.14. Haz un programa que dibuje un cuadrado de 21 × 21 (no considerar las columnas 22 y siguientes).

```
 10  FOR I = 0 TO 21
 20  PRINT AT 0, I; "*"
 30  NEXT I
 40  FOR J = 1 TO 20
 50  PRINT AT J, 0; "0"
 60  PRINT AT J, 21; "*"
 70  NEX J
 80  FOR K = 0 TO 21
 90  PRINT AT 21, K; "*"
100  NEXT K
110  END
```

— El bucle **10 - 30** imprime el lado superior del cuadrado.
— El bucle **40 - 70** imprime los lados izquierdo y derecho del cuadrado.
— El bucle **80 - 100** imprime la base del cuadrado.

Nota: En los microordenadores MSX hay que sustituir las líneas **20, 50, 60** y **90** por

```
20  LOCATE I, 0 : PRINT "*"
50  LOCATE 0, J : PRINT "*"
60  LOCATE 21, J : PRINT "*"
90  LOCATE K, 21 : PRINT "*"
```

13.15. Escribe un programa que dibuje las diagonales de un cuadrado de 21 × 21.

Se supone que los puntos (1, 1) y (21, 21) son dos vértices opuestos del cuadrado.

```
10 FOR I = 1 TO 21
20 PRINT AT I, I; "*"
30 PRINT AT 21 – I, 21 – I; "*"
40 NEXT I
50 END
```

La instrucción **20** imprime los puntos de una diagonal, y la **30** imprime los de la otra diagonal.

Nota: En los microordenadores MSX hay que sustituir las líneas **20** y **30** por

```
20 LOCATE I, I : PRINT "*"
30 LOCATE 21 – I, 21 – I : PRINT "*"
```

13.16. ¿Qué hace el siguiente programa?

```
10 FOR I = 0 TO 21
20 PRINT AT I, 0; "*"
30 PRINT AT I, 31; "*"
40 NEXT I
50 FOR J = 0 TO 31
60 PRINT AT 0, J; "*"
70 PRINT AT 21, J; "*"
80 NEXT J
90 END
```

El bucle **10 - 40** llena de asteriscos las columnas 0 y 31.

El bucle **50 - 80** llena de asteriscos las filas 0 y 21.

En conclusión, este programa dibuja un rectángulo de vértices (0,0), (0,31), (21,0) y (21,31).

Nota: En los microordenadores MSX hay que sustituir las líneas **20, 30, 60** y **70** por

```
20 LOCATE 0, I : PRINT "*"
30 LOCATE 31, I : PRINT "*"
60 LOCATE J, 0 : PRINT "*"
70 LOCATE J, 21 : PRINT "*"
```

PROGRAMAS RESUELTOS

13.1. ¿Qué instrucciones son erróneas? ¿Por qué?

a) **100 PRINT TAB (23); "HOLA"**
b) **200 PRINT TAB (10); 1; TAB (25); "N"**
c) **300 PRINT N; TAB (13); "ADIOS"**
d) **400 PRINT TAB (7); S : TAB (9); A$**

a) La instrucción **100** es errónea porque **TAB (23)** y **"HOLA"** están separados por un punto y coma. Deben estar separados por una coma.
b) La instrucción **200** es correcta.
c) La instrucción **300** es correcta.
d) En la línea **400** hay un error (falta la palabra **PRINT** después de los dos puntos).

13.2. Utiliza la instrucción **PRINT TAB** para resolver lo indicado en los apartados siguientes:

a) Imprimir la palabra MENÚ a partir de la columna 10.
b) Imprimir el número 3.1416 y el texto NUMERO PI a partir de las columnas 5 y 15, respectivamente.

a) **10 PRINT TAB (10); "MENU"**
b) **10 PRINT TAB (5); 3.1416; TAB (15); "NUMERO PI"**

13.3. Identifica los errores contenidos en el siguiente programa:

```
10  INPUT A$
20  LET L = LEN (A$)
30  PRINT AT L, L, A$
40  IF L = 1 THEN GOTO 10
50  PRINT AT L; "FIN"
60  END
```

— En la instrucción **30,** entre L y A$, tiene que haber un punto y coma en lugar de una coma.
— En la instrucción **50** falta una coordenada.

13.4. Haz un programa que vaya admitiendo números y tabule la pantalla para que en la columna 10 aparezcan los números que se van introduciendo y

en la columna 20 el número más pequeño introducido hasta este momento.

```
10 PRINT TAB (10); "ULTIMO"; TAB (20); "MAS PEQUEÑO"
20 INPUT X
30 LET P = X
40 PRINT TAB (10); X; TAB (20); P
50 INPUT X
60 IF X < P THEN LET P = X
70 GOTO 40
80 END
```

13.5. Elabora un programa que imprima a partir de la columna 5 el nombre de una persona; a partir de la 20, su edad y a partir de la 25, su peso. Particularizar el programa que lea los nombres, edades y pesos de seis personas mediante **READ** y **DATA**.

```
 10 PRINT TAB (5); "NOMBRE"; TAB (20); "EDAD"; TAB (25);
    "PESO"
 20 FOR I = 1 TO 6
 30 READ N$, E, P
 40 PRINT TAB (5); N$; TAB (20); E; TAB (25); P
 50 NEXT I
 60 DATA "MARIA", 15, 43
 70 DATA "JUAN", 16, 55
 80 DATA "ERNESTO", 13, 37
 90 DATA "ISABEL", 16, 47
100 DATA "ELENA", 15, 39
110 DATA "MIGUEL", 14, 61
120 END
```

13.6. Escribe un programa que lea un texto y lo imprima en diagonal; es decir, el primer carácter en la fila 0, columna 0; el segundo carácter en la fila 1, columna 1, y así sucesivamente.

```
10 INPUT T$
20 LET L = LEN (T$)
30 IF L > 21 THEN PRINT "DEMASIADO LARGO" : GOTO 10
40 CLS
50 FOR I = 1 TO L
60 LOCATE I – 1, I – 1 : PRINT MID$ (T$, I, 1)
70 NEXT I
80 END
```

Nota: En los microordenadores ZX Spectrum, sustituir la línea **60** por

```
60 PRINT AT I – 1, L – 1; T$ (I TO I).
```

13.7. Haz un programa que dibuje la letra I con asteriscos, a partir de la columna 6, con una altura de diez puntos y con una anchura de tres.

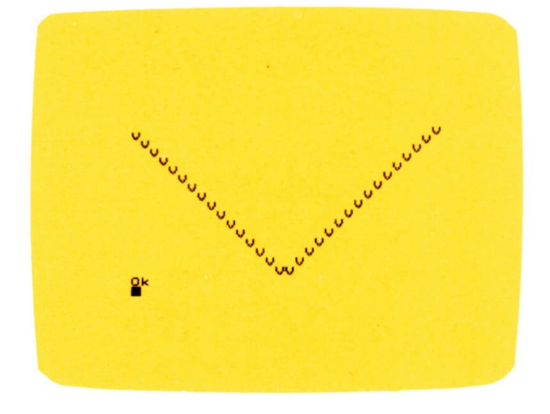
```
10  CLS
20  FOR I = 1 TO 10
30  PRINT TAB (6); "***"
40  NEXT I
50  END
```

13.8. Haz un programa que dibuje la letra A con las mismas condiciones que en el programa anterior.

```
 10  CLS
 20  PRINT TAB (6); "***"
 30  FOR I = 1 TO 3
 40  PRINT TAB (6); "*  *"
 50  NEXT I
 60  PRINT TAB (6); "***"
 70  FOR I = 1 TO 5
 80  PRINT TAB (6); "*␣*"
 90  NEXT I
100  END
```

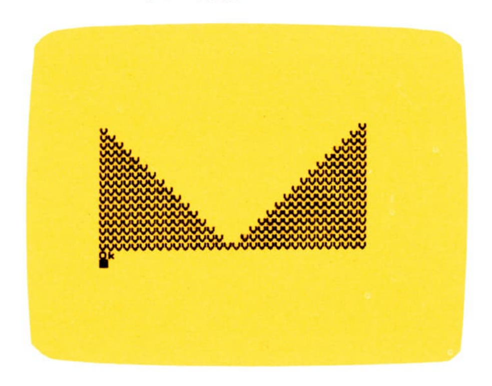
13.9. Por medio de la instrucción **PRINT TAB,** haz un programa que dibuje una V que ocupe toda la pantalla, imprimiendo uves.

```
10  CLS
20  FOR I = 0 TO 15
30  PRINT TAB (I); "V"; TAB (31 – I); "V"
40  NEXT I
50  END
```



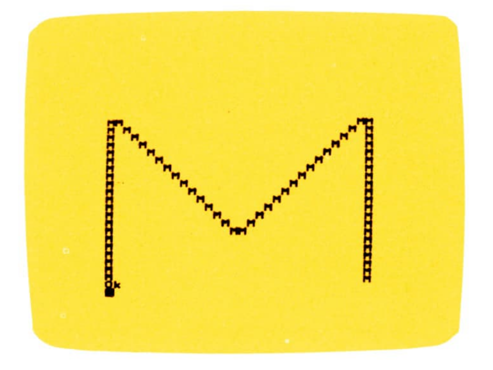
13.10. Diseña un programa que rellene la parte inferior de la uve de la pantalla del ejercicio anterior con uves.

```
10  CLS
20  LET A$ = "V"
30  FOR I = 0 TO 15
40  PRINT A$: TAB (31 – I); A$
50  LET A$ = A$ + "V"
60  NEXT I
70  END
```



13.11. Realiza un programa que dibuje una M que ocupe toda la pantalla.

```
10  CLS
20  FOR I = 1 TO 15
30  PRINT "M"; TAB (I); "M"; TAB (31 – I); "M"; TAB (31); "M"
40  NEXT I
50  FOR I = 16 TO 21
60  PRINT "M"; TAB (31); "M"
70  NEXT I
80  END
```



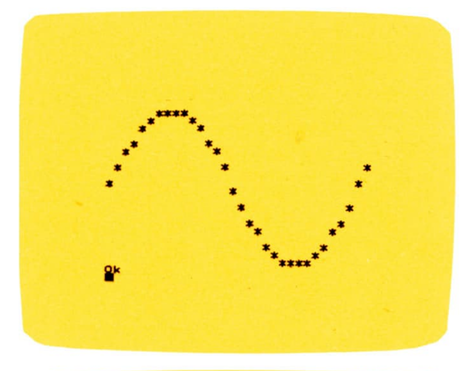
13.12. El siguiente programa dibuja la función seno en una pantalla con baja resolución.

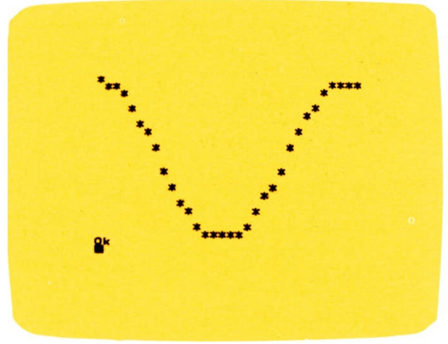
```
10 FOR I = 0 TO 31
20 PRINT AT 10 – INT (10 * SIN (I * PI / 15)), I; "*"
30 NEXT I
```

Modifica este programa para que dibuje el coseno.

Basta con sustituir la función seno por la función coseno.

```
10 FOR I = 0 TO 31
20 PRINT AT 10 – INT (10 * COS (I * PI / 15)), I; "*"
30 NEXT I
```





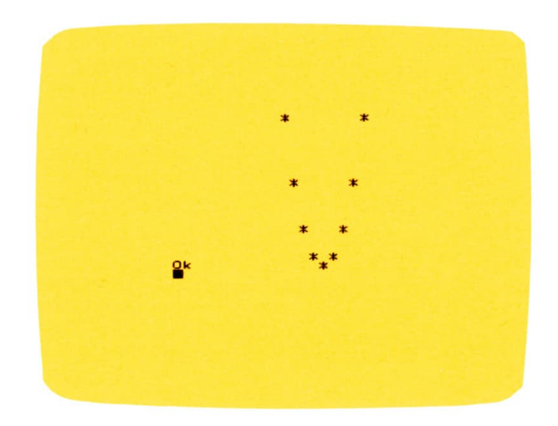
Nota: En los microordenadores MSX, los programas anteriores deben ser sustituidos por los siguientes:

```
 1 REM SENO
10 CLS
20 FOR I = 0 TO 31
30 LOCATE I, 10 – INT (10 * SIN (I * 3.1416/15)), PRINT "*"
40 NEXT I
50 LOCATE 0, 21
60 END
```

```
 1 REM COSENO
10 CLS
20 FOR I = 0 TO 31
30 LOCATE I, 10 – INT (10 * COS (I * 3.1416/15)) : PRINT "*"
40 NEXT I
50 LOCATE 0, 21
60 END
```

13.13. Haz un programa que dibuje la función F(X) = X^2 utilizando la instrucción **PRINT AT.**

```
10 FOR I = –4 TO 4
20 PRINT AT 21 – I * I, 15 + I; "*"
30 NEXT I
```

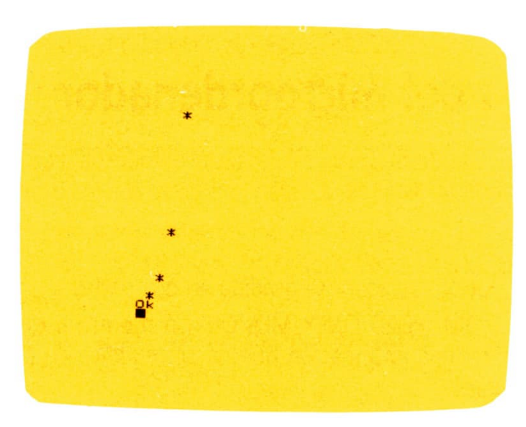


Nota: En los microordenadores MSX, el programa debe ser escrito como sigue:

```
10 CLS
20 FOR I = –4 TO 4
30 LOCATE 15 + I, 21 – I ↑ 2 : PRINT "*"
40 NEXT I
50 LOCATE 0, 21
60 END
```

13.14. Diseña un programa que por medio de la instrucción PRINT AT dibuje la función F(X) = e^x.

```
10 FOR I = 0 TO 3
20 PRINT AT 21 – EXP (I), I; "*"
30 NEXT I
```



Nota: En los microordenadores MSX, el programa anterior debe escribirse así:

```
10  CLS
20  FOR I = 0 TO 3
30  LOCATE I + 1, 21 - EXP (I) : PRINT "*"
40  NEXT I
50  LOCATE 0, 21
60  END
```

14 La memoria del microordenador

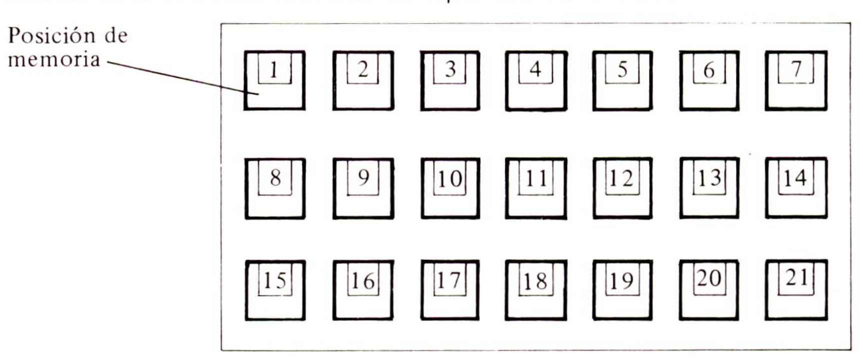
Memorias RAM y ROM

La memoria de un microordenador puede considerarse dividida en dos partes:

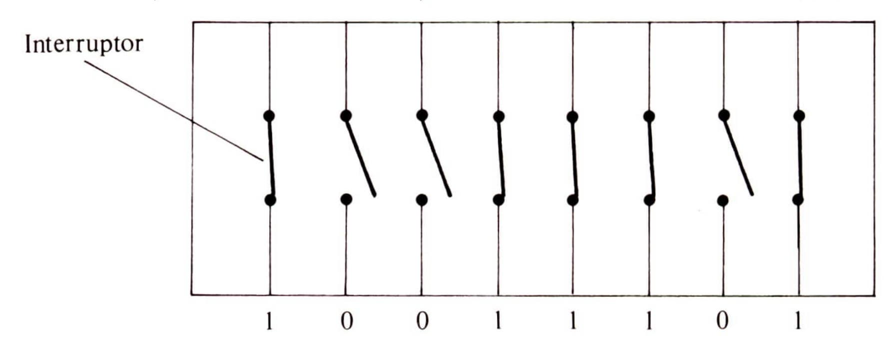
— Una parte contiene la memoria **ROM** (READ ONLY MEMORY) o **memoria de sólo lectura.** En esta parte se almacena información que no se modifica en el proceso de ejecución de un programa.

— La otra parte contiene la memoria **RAM** (RANDOM ACCESS MEMORY) o **memoria de acceso aleatorio.** En esta memoria se puede leer y almacenar información, y ésta desaparece cuando se enchufa el microordenador. En la memoria RAM se almacenan el programa BASIC y los datos que se introducen con **INPUT** o con **READ - DATA.**

Ambos tipos de memoria pueden imaginarse como una larga lista de casillas numeradas. Cada una de las casillas se llama **posición de memoria.**



A su vez, cada posición de memoria puede considerarse como un conjunto de ocho interruptores colocados en fila, pudiendo estar cada uno abierto o cerrado.



Un interruptor cerrado representa un 1, y un interruptor abierto, un 0.

Cada cero o uno se llama **bit** (dígito binario) y la sucesión de los 8 bits se llama **byte** (1 byte comprende 8 bits).

Toda la información que se maneja en el microordenador debe estar formada por unos y ceros; es decir, debe estar en el **sistema binario.**

Así, para representar las letras, los dígitos, los signos de puntuación y los símbolos de las operaciones aritméticas debe utilizarse una secuencia de ocho ceros y unos.

En la mayoría de los microordenadores se ha seguido un mismo convenio para asignar en su memoria un número binario a cada carácter. Este convenio constituye el llamado código ASCII (American Standar Code for Information Interchange). Este código se transcribe en el Apéndice A, en el cual, cada número binario ha sido sustituido por su equivalente número en base decimal.

RESUMEN DE LAS INSTRUCCIONES

PEEK

Esta instrucción extrae el contenido de una posición determinada de memoria.



Ejemplo

La instrucción

10 PRINT PEEK 50323

se imprime en la pantalla un número decimal comprendido entre 0 y 255, que es el contenido de la posición de memoria cuya dirección viene dada por el número 50323.

El contenido de una posición de memoria puede estar comprendido entre los números decimales 0 y 255, que corresponden a los binarios 00000000 y 11111111.

POKE

Esta instrucción almacena en una posición determinada de memoria un número comprendido entre 0 y 255.

Ejemplo

La instrucción

```
10 POKE 50323, 65
```

almacena en la posición de memoria **50323** un conjunto de ceros y unos, correspondientes al número decimal 65.

Observaciones

Esta instrucción sólo actúa sobre la memoria RAM.

Se debe tener cuidado al elegir las posiciones de memoria sobre las que se actúa, ya que puede perturbarse el funcionamiento del microordenador. Para evitar problemas, debe consultarse el manual de instrucciones del microordenador correspondiente.

ASC (O CODE)

Esta instrucción da el código numérico correspondiente a un carácter.



Ejemplo

El programa

```
10 LET A$ = "ANIBAL"
20 PRINT ASC (A$)
```

imprime en la pantalla el número 65, código ASCII del primer carácter de la cadena A$, que es la letra A (mayúscula).

CHR$

Esta instrucción da el símbolo correspondiente a un número en el código ASCII.



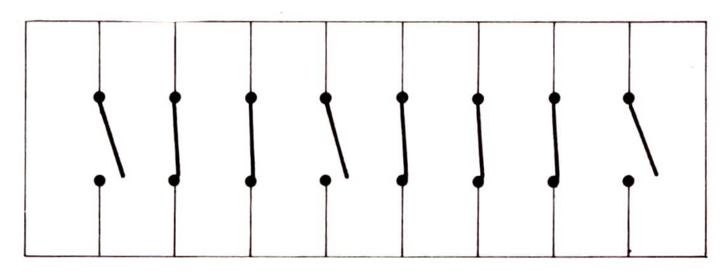
Ejemplo

La instrucción

```
10 PRINT CHR$ (65)
```

imprime en la pantalla la letra A correspondiente al número 65 del código ASCII.

14.1. Indica el valor del byte correspondiente a una posición de memoria cuyos interruptores se encuentran en el estado indicado en la figura.

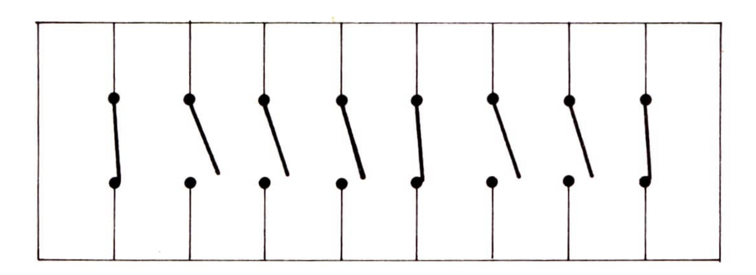


Cada interruptor abierto equivale a un cero, y cada interruptor cerrado, a un uno. Luego el número binario incluido en esa posición de memoria es

01101110.

14.2. Representa en un esquema similar al anterior el valor del byte representado por esta secuencia binaria: 10001001.

Siguiendo el criterio de que un interruptor abierto es 0 y uno cerrado es 1, el conjunto de ocho interruptores debe estar dispuesto como se indica en este esquema.

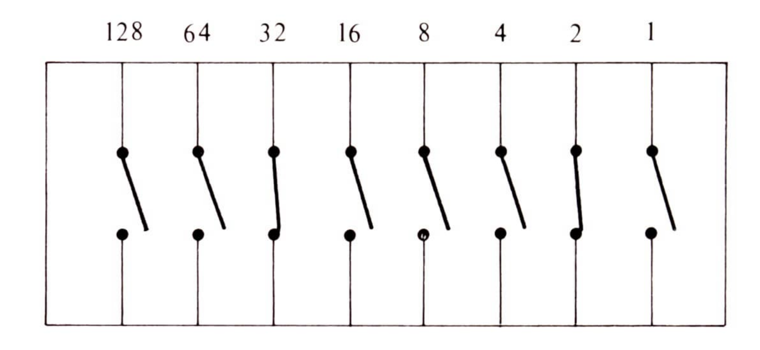


14.3. Ayudándote de un esquema con circuitos como el que venimos utilizando, representa mediante una secuencia de ocho bits el número 34.

En un byte, los bits pueden numerarse del 0 al 7, de derecha a izquierda. Cada uno de ellos si es 1, representa un valor doble del que corresponde al que se encuentra inmediatamente a su derecha.

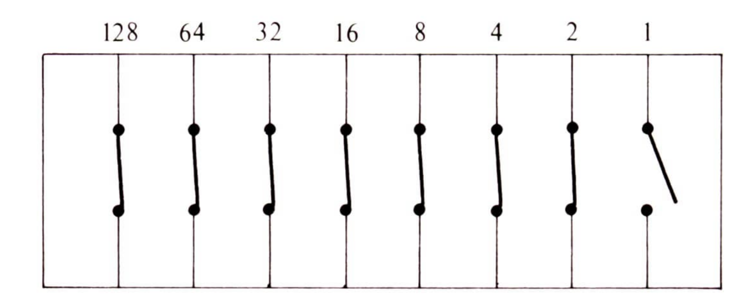
Así, el bit 0 representa el valor 1; el bit 1, el valor 2; el bit 2, el valor 4; el bit 3, el valor 8; el bit 4, el valor 16; el bit 5, el valor 32; el bit 6, el valor 64; y el bit 7, el valor 128.

Así, para representar 34 deben estar cerrados el segundo y el quinto bit; luego el número binario es 00100010.



14.4. Escribe una secuencia binaria cuyo valor corresponda a 254. En esta secuencia, ¿qué bits es cero, el último de la derecha o el último de la izquierda?

$254 = 128 + 64 + 32 + 16 + 8 + 4 + 2 + 0 = 1 \cdot 2^7 +$
$+ \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 +$
$+ 2^0$



Así, la secuencia binaria correspondiente a 254 es 11111110.

14.5. Halla la representación en base decimal de los siguientes bytes:

00000001
00000010
00000011
00000100
00000101

$0 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 +$
$+ 1 \cdot 2^0 = 1$

Luego, el número 1 en binario es 00000001.

De la misma manera se obtienen las representaciones binarias de los otros números escritos en base decimal.

El número 2 en binario es 00000010.

$2 = 0 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + + 0 \cdot 2^0$

El número 3 en binario es 00000011.

$3 = 0 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + + 1 \cdot 2^1 + 1 \cdot 2^0$

El número 8 en binario es 00001000.

$8 = 0 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + + 0 \cdot 2^1 + 0 \cdot 2^0$

Finalmente, el número 9 en binario es 00001001.

$9 = 0 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$

14.6. Encuentra todas las secuencias binarias comprendidas entre 11111001 y 11111111.

Las dos únicas posibilidades se obtienen en colocar un cero y un uno, alternándolos, en las posiciones 2 y 3:
11111011
que corresponde a 247
y
11111101
que corresponde a 254.

14.7. Haz un programa que imprima las representaciones en base decimal de los números binarios 11110001, 01101101, 01010101, 10101010, 01111110 y 10000001.

```
1 REM PASO DE BINARIO A DECIMAL
5 FOR J = 1 TO 6
7 LET N = 0 : LET P = 0
10 READ A$(J)
20 FOR I = LEN (A$) TO 1 STEP -1
30 LET B$ = MID$ (A$, LEN(A$) - P, 1)
40 LET N = N + VAL (B$) * 2 ↑ P
50 LET P = P + 1
60 NEXT I
70 PRINT A$; " EN BASE DECIMAL SE EXPRESA ASI: "; N
80 NEXT J
90 END
100 DATA 11110001, 01101101, 01010101, 10101010,
    01111110, 10000001.
```

— Para cada valor de J se lee uno de los seis números binarios.
— La instrucción **30** extrae cada uno de estos dígitos binarios y los almacena temporalmente en la variable de cadena B$.
— La instrucción **40** pasa a valor numérico lo que almacena en B$, multiplica por la correspondiente potencia de 2 y almacena el resultado en N. Esto lo hace para cada dígito binario, y el concluirse la ejecución del bucle **20 - 60,** N almacena el número en base decimal.

Nota: En el microordenador ZX Spectrum debe sustituirse la instrucción **30** por

```
30 LET B$ = A$ (LEN(A$) – P TO LEN(A$) – P)
```

14.8. Halla la representación en binario de los números naturales 0, 1, 2, ..., 9.

Teniendo en cuenta los valores relativos de cada bit, se obtiene lo siguiente:

	128	64	32	16	8	4	2	1
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	1	0
3	0	0	0	0	0	0	1	1
4	0	0	0	0	0	1	0	0
5	0	0	0	0	0	1	0	1
6	0	0	0	0	0	1	1	0
7	0	0	0	0	0	1	1	1
8	0	0	0	0	1	0	0	0
9	0	0	0	0	1	0	0	1

14.9. Halla la representación en binario del número 248.

Se divide sucesivamente por 2 y escribimos de izquierda a derecha el último cociente seguido de los sucesivos restos.

```
248 |2
04   124 |2
 08   04   62|2
 0     0  02  31|2
           0  11  15|2
               1   1  7 |2
                      1  3 |2
                         1  1
```

Se obtiene 11111000 que corresponde al 248.

14.10. Construir un programa que imprima en la pantalla la representación en binario de los números 245 a 255.

```
5 REM PASO DE DECIMAL A BINARIO
10 FOR J = 1 TO 11
15 LET B$ = " "
20 READ N
30 LET I = INT (N/2)
40 LET B = N - 2 * I
50 IF B = 0 THEN LET B$ = 0 + B$ : GOTO 70
60 LET B$ = "1" + B$
70 LET N = I
80 IF N = 0 THEN GOTO 100
90 GOTO 30
100 PRINT "EL NUMERO"; N; "EN BINARIO ES:"; B$
105 NEXT J
110 END
200 DATA 245, 246, 247, 248, 249, 250, 251, 252, 253, 254,
    255
```

— Para cada valor de J se lee un número en base decimal; previamente, en la variable B$ se almacena un espacio en blanco.

— Las instrucciones **30** y **40** calculan los restos sucesivos de la división de N por 2. Por ejemplo, si N = 245, en la primera división se obtiene el siguiente resto:

```
30 LET I = INT (245/2) = INT (122.5) = 122
40 LET B = 245 - 2 * 122 = 245 - 244 = 1
```

— La instrucción **50** añade por la izquierda un 0 a la cadena almacenada en B$, si se cumple la condición B = 0, y la **60** añade un 1 en caso de no cumplirse B = 0.

— La instrucción **70** guarda en N el siguiente cociente parcial (en el ejemplo es 122) con el cual se realizará otra división por 2, y así hasta que N sea igual a 0, condición que comprueba la instrucción **80.** En el caso de que la condición se cumpla, se transfiere el control a la instrucción **100,** que imprime el número en binario.

14.11. Explica lo que hace el siguiente programa:

```
10 FOR I = 0 TO 9
20 POKE 34230 + I, I
30 NEXT I
40 END
```

Almacena en las posiciones de memoria 34230, 34231, ..., 34239, los equivalentes números en binario de los números es-

critos en base decimal, que van del 0 al 9. Estos valores los va tomando la variable I del bucle.

14.12. Haz un programa que almacene en posiciones de la memoria de dirección consecutiva las letras mayúsculas A, B, C, D, E y F.

```
10 FOR I = 0 TO 5
20 POKE 50000 + I, 65 + I
30 NEXT I
40 END
```

Este programa almacena en la posición de memoria **50000** y las cinco siguientes, los códigos ASCII de las letras A (65 + 0), B (65 + 1), C (65 + 2), D (65 + 3), E (65 + 4) y F (65 + 5).

(Antes de introducir valores en una posición de memoria mediante la instrucción **POKE,** debe consultarse el manual de instrucciones en el que vendrá incluido un mapa de la memoria del microordenador.)

14.13. Indica qué imprimiría en la pantalla el siguiente programa:

```
10 FOR I = 1 TO 5
20 READ N
30 LET P = PEEK (N)
40 PRINT P
50 NEXT I
60 DATA 42627, 43700, 43701, 43702, 44800
70 END
```

Imprime el contenido de las posiciones de memoria **42627, 43700, 43701, 43702** y **44800**. (Los números contenidos en estas posiciones los escribe en base decimal.) Los números obtenidos deben estar comprendidos entre 0 y 255.

14.14. Escribe un programa que imprima en la pantalla los números del códigos ASCII correspondientes a las letras A, a, B, b, c y C.

```
 10 FOR I = 1 TO 6
 20 READ A$
 30 PRINT ASC (A$)
 40 NEXT I
 50 END
100 DATA A, a, B, b, c, C.
```

El programa lee cada una de las letras contenidas en la instrucción **100 DATA,** las almacena en A$ e imprime en la pantalla su código ASCII.

(En el ZX Spectrum debe modificarse la instrucción **30** por **30 CODE A$**.)

14.15. Fijándote en los valores obtenidos en el ejercicio anterior, ordena de anterior a posterior (<) las palabras siguientes: Bernardo, azucena, Antonio, abeto, canica, Claudio.

Antonio < Bernardo < Claudio < abeto < azucena < canica.

Las letras mayúsculas y minúsculas poseen un orden alfabético. (Pero como las mayúsculas tienen códigos ASCII menores, las palabras cuyas iniciales son mayúsculas aparecen antes que las palabras cuyas iniciales son letras minúsculas.)

14.16. Haz un programa que imprima en la pantalla los caracteres cuyos números ASCII son 56, 57, 58, 59 y 60.

```
10 FOR I = 56 TO 60
20 PRINT CHR$ (I)
30 NEXT I
40 END
```

La variable I toma los valores de los códigos ASCII, que van del **56** al **60,** y en la instrucción **20** imprime el carácter correspondiente.

PROGRAMAS RESUELTOS

14.1. Escribe lo que imprimiría el siguiente programa al ser ejecutado:

```
 5 LET P = 0 : LET N = 0
10 READ A$
20 FOR I = LEN (A$) TO 1 STEP –1
30 LET B$ = MID$ (A$, LEN (A$) – P, 1)
40 LET N = N + BAL(B$) * 2 ↑ P
50 LET P = P + 1
60 NEXT I
70 PRINT A$; " EQUIVALE A "; N
80 GOTO 5
90 DATA "10101010", "01010101", "11111000", "00011111"
```

Escribe los equivalentes números escritos en base decimal de los números binarios incluidos en la instrucción **DATA.**

Así, en la pantalla aparecen estos resultados



Se emite, además, un mensaje de error al acabarse los datos.

14.2. El siguiente programa pasa a sistema binario, números expresados en base decimal. Explica, instrucción a instrucción, su funcionamiento.

```
 10 READ N
 20 FOR I = 1 TO N
 30 LET B$ = " "
 40 READ A
 50 LET K = INT (A/2)
 60 LET B = A − 2 * K
 70 IF B = 0 THEN LET B$ = "0" + B$ : GOTO 90
 80 LET B$ = "1" + B$
 90 LET A = K
100 IF A = 0 THEN GOTO 120
110 GOTO 50
120 PRINT A; "EN BINARIO ES:"; B$
130 NEXT I
135 DATA 5
140 DATA 25, 125, 50, 200, 10
150 END
```

— Para cada valor de I se lee un número escrito en base decimal (hay cinco, de ahí que el número de lecturas debe coincidir con este valor de N). Previamente, en la variable B$ se almacena un espacio en blanco.

— Las instrucciones **50** y **60** calculan los restos sucesivos de la división de N por 2.

— La instrucción **70** añade por la izquierda un 0 a la cadena almacenada en B$, si se cumple la condición B = 0, y la **80** añade un 1 en caso de no cumplirse B = 0.

— La instrucción **70** guarda en A el siguiente cociente parcial, con el cual se realizará otra división por 2, y así hasta que A sea igual a 0, condición que comprueba la instrucción **120** que imprime el número en binario.

14.3. Explica qué hace la siguiente instrucción:

```
17 POKE 25328, 29
```

Coloca en la posición de memoria 25328 el equivalente en binario al número 29.

14.4. Introduce en la memoria los diez primeros números naturales impares en posiciones de memoria, tal que sus direcciones sean consecutivas, empezando por la de dirección **45000**.

```
10 FOR I = 0 TO 9
20 POKE 45000 + I, 2 * I + 1
30 NEXT I
40 END
```

14.5. Explica qué se obtiene en la pantalla si se escribe la siguiente expresión:

```
PRINT PEEK (45005)
```

Imprime el contenido de la posición de memoria **45005.**

14.6. Haz un programa que verifique si en las posiciones de memoria indicadas en el ejercicio 4 se encuentran almacenados los diez primeros números naturales impares.

```
10 FOR I = 0 TO 9
20 PRINT PEEK (45000 + I)
30 NEXT I
```

14.7. Explica la función que cumple la instrucción **ASC** (o **CODE**).

Permite obtener el código ASCII del primer carácter de una cadena.

14.8. Indica qué se obtiene en la pantalla al ejecutar el siguiente programa:

```
10 READ C$
20 LET A = ASC (C$)
30 PRINT A
40 GOTO 10
50 DATA " ", "<", "=", ">"
60 END
```

Se obtienen estos resultados:

```
32
60
61
62
```

que son los códigos ASCII del espacio y los signos <, = y >.
(Al acabarse los datos aparecerá un mensaje de error.)

14.9. Escribe un programa que halle el número del código ASCII correspondiente a las letras I y L, y según el resultado obtenido después de ejecutar el programa, indica qué símbolo pondrías entre las siguientes palabras:

Luis ... laberinto

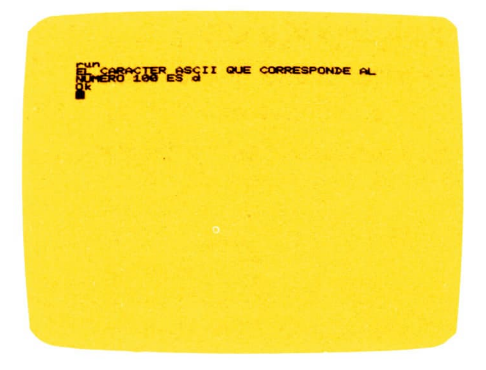
```
10 PRINT ASC ("Luis")
20 PRINT ASC ("laberinto")
```

Luis < laberinto

14.10. Indica qué se obtiene en la pantalla cuando se ejecuta el siguiente programa. (Consulta el código ASCII en el apartado 7 de este capítulo.)

```
10 READ N
20 PRINT "EL CARACTER ASCII QUE CORRESPONDE AL
   NUMERO"; N; "ES"; CHR$ (N)
30 DATA 100
40 END
```

Se obtiene este resultado:



El carácter que corresponde al número 100 es 5.

14.11. Realiza un programa que haga aparecer en la pantalla los caracteres del código ASCII correspondientes a los números 33 a 125, dispuestos en la siguiente forma:

NUMERO	CARACTER
33	.
.	.
.	.
.	.

```
10 PRINT "NUMERO", "CARACTER"
20 FOR I = 32 TO 125
30 PRINT I, CHR$ (I)
40 NEXT I
```

Apéndice

Palabras BASIC	*Función que realiza*	*Página del libro*
ABS (X)	Da el valor absoluto del número X.	145
ASC (X$)	Da el número de código ASCII correspondiente al primer carácter de la cadena X$.	210
AT F, C	Sitúa el cursor en la posición correspondiente a la fila F y a la columna C.	183
ATN (X)	Calcula el arco cuya tangente es X. (El arco calculado se expresa en radianes.)	155
CHR$ (X)	Da el carácter del código ASCII correspondiente al número X. (X es un entero comprendido entre 32 y 128.)	210
CLS	Borra lo escrito en la pantalla, pero no lo almacenado en la memoria del microordenador.	26
C LOAD "XXX" (o LOAD "XXX")	Carga en la memoria del microordenador un programa grabado en una casete con el nombre XXX	216
CONT	Continúa la ejecución de un programa interrumpido por la instrucción **STOP.**	62
CODE (X$)	Da el número del código ASCII correspondiente al primer carácter de la cadena X$.	210
COS (X)	Calcula el coseno del ángulo X, expresado en radianes.	155
C SAVE "XXX" (o SAVE "XXX")	Graba en una casete con el nombre XXX un programa almacenado en la memoria del microordenador.	216
DATA	Almacena los datos que serán leídos por una instrucción **READ.**	69
DEF FNA (X)	Define una función numérica de nombre FNA y de variable X. (Para definir otras funciones cambiar la letra A por otras del abecedario.)	155
DIM L(N) DIM T (N,M)	Dimensiona una variable de un índice. Dimensiona una variable de dos índices.	114 125
END	Da por finalizado un programa.	20

ENTER	Hace que el dato o instrucción tecleados se almacene en la memoria.	20
EXP (X)	Calcula la potencia e^x, siendo e = 2.71828	155
FNA (X)	Da el valor de la función de nombre FNA previamente definida, para el valor X.	155
FOR V = i TO t STEP s	Ejecuta un conjunto de instrucciones, llamado ***bucle,*** que empieza con la instrucción FOR y termina con una instrucción NEXT. (Variable: V; valor inicial de V, i; tope, t; incremento, s.)	78
GOSUB n	Transfiere el control de la ejecución de una subrutina que empieza en una instrucción numerada con n.	164
GOTO n	Transfiere el control de la ejecución a la instrucción del programa numerado con n. Esta instrucción puede ser anterior o posterior a n.	48
IF (condición) THEN (instrucciones)	Si se cumple la ***condición*** entonces se ejecutan las instrucciones que siguen a THEN; y si no se cumple se ejecuta la instrucción escrita en la línea siguiente.	51
INPUT A INPUT A$	Permite introducir datos numéricos o alfanuméricos, los cuales se asignan a la variable A o A$, respectivamente.	33 94
INT (X)	Calcula la parte entera del número X.	145
LEFT $ (A$,N)	Extrae los N primeros caracteres de la cadena almacenada en A$.	100
LEN (A$)	Da el número de caracteres de la cadena almacenada en A$.	97
LET variable = = expresión	Calcula el valor de la expresión y lo asigna a la variable.	20
LIST LIST n LIST n_1-n_2 LIST n –	Hace aparecer en la pantalla: — Todo el listado del programa almacenado en la memoria. — La instrucción n. — Las instrucciones comprendidas entre n_1 y n_2, ambas inclusive. — Las instrucciones comprendidas entre n y la última, ambas inclusive.	24

LN (X)	Calcula el logaritmo natural o neperiano de X.	**155**
LOCATE C, F	Sitúa el cursor en la posición correspondiente a la columna C y a la fila F.	**186**
LOG (X)	Calcula el logaritmo decimal de X.	**155**
MID $ (A$,P,N)	Extrae N caracteres de la cadena almacenada en A$ desde la posición P, siguiendo hacia la derecha.	**100**
NEW LINE	Hace que el dato o instrucción tecleados se almacenen en la memoria.	**20**
NEW	Borra todo el contenido de la memoria.	**26**
NEXT V	Transfiere la ejecución del programa a la instrucción FOR del bucle siempre que el valor de V sea menor o igual que el tope del bucle.	**79**
ON V GOSUB $n_1 \ldots n_k$	Transfiere el control de la ejecución a las subrutinas que se inician con las instrucciones n_1, n_2, ..., n_k, según V valga 1, 2, ..., k.	**169**
ON V GOTO $n_1 \ldots n_k$	Transfiere el control de la ejecución a las instrucciones n_1, n_2, ..., n_k del programa según V valga 1, 2, ..., k, respectivamente.	**169**
PAUSE N	Detiene la ejecución del programa y retiene la imagen un tiempo que depende de N.	**105**
PEEK (X)	Da el contenido de la posición de la memoria de dirección X.	**206**
POKE N, V	Almacena el valor de V en la posición de la memoria de dirección N (V es un entero comprendido entre 0 y 255).	**206**
PRINT	Escribe o imprime el valor de la variable, el resultado de operaciones aritméticas y texto entrecomillado.	**36**
RANDOMIZE y RND	Da un número aleatorio entre 0 y 1.	**150**
READ A, B, ..., X	Asigna a las variables A, B, ..., X los datos almacenados en una instrucción **DATA,** en forma secuencial.	**69**
REM	Permite añadir aclaraciones o comentarios en un programa sin afectar su ejecución.	**35**

RESTORE	Permite volver a leer desde el principio de los datos contenidos en las instrucciones **DATA.**	72
RETURN (tecla)	Hace que el dato o instrucción tecleado se almacene en la memoria.	20
RETURN	Última instrucción de una subrutina, cuya función es transferir el control de la ejecución siguiente a la GOSUB del programa principal.	164
RIGHT$ (A$,N)	Extrae los N últimos caracteres de la cadena almacenada en A$.	100
RND (A)	Calcula un número aletorio entre 0 y 1 (A es un argumento, al que habitualmente se asigna el valor 1.)	149
RUN	Inicia la ejecución del programa. ("Correr el programa".)	21
SGN (X)	Calcula el signo de X (+1, si X > 0; 0, si X = 0; −1, si X < 0).	145
SIN (X)	Calcula el seno del ángulo X, expresado en radianes.	155
SQR (X)	Calcula la raíz cuadrada de X.	155
STOP	Detiene la ejecución del programa justo en la línea en que se encuentra STOP.	61
STR$(N)	Convierte el número almacenado en N en una cadena numérica.	98
TAB (N)	Mueve la posición del cursor a la columna N.	179
TAN (X)	Calcula la tangente de X, expresado en radianes.	155
TO	A$ (N TO M): extrae una subcadena de A$ desde la posición N a la M.	102
VAL (N$)	Convierte la cadena numérica almacenada en N$ en su valor numérico.	98
WAIT (N)	Detiene la ejecución del programa y retiene la imagen de la pantalla un tiempo, que depende de N.	105

COLECCIÓN BASIC

Basic Programación

Gráficos, Colores y Música en el ZX Spectrum

MSX. Programación con Gráficos, Colores y Música

Programas resueltos en BASIC

Programas de aplicaciones en BASIC

distribuidor exclusivo cesma sa
C/ Aguacate, 25
28044 MADRID